Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

ДОПУСТИТИ ДО ЗАХИСТУ:
В.о. завідувача кафедри
кібербезпеки та захисту інформації
_____ Іван ПАРХОМЕНКО
«____» червня 2023 р.

ПОЯСНЮВАЛЬНА ЗАПИСКА

кваліфікаційної роботи

галузь знань 12 Інформаційні технології
(шифр і назва галузі знань)

спеціальність 125 Кібербезпека
(код і назва спеціальності)

освітній ступень бакалавр

освітня програма Кібербезпека
(назва освітньо-професійної програми)

на тему: Засоби підвищення захищеності сучасних блокчейн додатків

Виконавець: студент IV курсу, групи КБ-41

_____ **Юрій ХАРЛАМОВ**
(підпис) (ім'я, прізвище)

	Ім'я, прізвище	**Підпис**
Керівник	Юрій ЩЕБЛАНІН	

Нормоконтроль	Олена БОГУСЛАВСЬКА	

Київ 2023

Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

ЗАТВЕРДЖЕНО:
В.о. завідувача кафедри кібербезпеки
та захисту інформації
_____ Сергій ТОЛЮПА
«24» жовтня 2022 р.

ЗАВДАННЯ
на виконання кваліфікаційної роботи

спеціальності 125 Кібербезпека
(код і назва спеціальності)

освітньої програми Кібербезпека
(назва освітньо-професійної програми)

Студенту **КБ-41** **Юрію Сергійовичу Харламову**
(група) (прізвище ім'я по батькові)

Тема кваліфікаційної роботи Засоби підвищення захищеності сучасних блокчейн Додатків

1. ПІДСТАВИ ДЛЯ ПРОВЕДЕННЯ РОБОТИ

Тема кваліфікаційної роботи затверджена на засіданні кафедри кібербезпеки та захисту інформації протокол №3 від 20.10.2022 р.

2. ВИХІДНІ ДАНІ ДЛЯ ПРОВЕДЕННЯ РОБІТ

Додатки засновані на розумних контрактах з використанням технології Ethereum, вразливості розумних контрактів.

3. ЗМІСТ РОЗРАХУНКОВО-ПОЯСНЮВАЛЬНОЇ ЗАПИСКИ

Нормативно-правова база у сфері захисту інформаційно-телекомунікаційних систем, проведення аналізу основних загроз для телекомунікаційних систем, проведення аналізу вразливостей додатків з використанням технології блокчейн та розробка рекомендацій щодо підвищення їх рівня захищеності

4. ВИМОГИ ДО РЕЗУЛЬТАТІВ ВИКОНАННЯ РОБОТИ

Практична цінність Підвищення рівня захищеності блокчейн додатків

5. ДАТА ВИДАЧІ ЗАВДАННЯ

Дата видачі завдання: 24 жовтня 2022 року

Завдання видав _____ (підпис) Юрій ЩЕБЛАНІН (ім'я, прізвище)

Завдання прийняла до виконання _____ (підпис) Юрій ХАРЛАМОВ (ім'я, прізвище)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Найменування етапів робіт	Строки виконання робіт (початок-кінець)	Відмітка про виконання
1	Уточнення постановки задачі	24.10.2022 – 22.01.2023	*виконано*
2	Аналіз літератури	29.01.2023 – 11.02.2023	*виконано*
3	Огляд технології блокчейн	12.02.2023 – 15.02.2023	*виконано*
4	Збір відомостей щодо технології Ethereum	16.02.2023 – 04.03.2023	*виконано*
5	Дослідження основних атак на розумні контракти	05.03.2023 – 21.03.2023	*виконано*
6	Аналіз інцидентів	22.03.2023 – 08.04.2023	*виконано*
7	Формування моделі загроз	09.04.2023 – 10.05.2023	*виконано*
8	Формування рекомендацій з підвищення безпеки	11.05.2023 – 27.05.2023	*виконано*
9	Підготовка до захисту кваліфікаційної роботи	28.05.2023 – 12.06.2023	*виконано*

Завдання видав _____ (підпис) Юрій ЩЕБЛАНІН (ім'я, прізвище)

Завдання прийняв до виконання _____ (підпис) Юрій ХАРЛАМОВ (ім'я, прізвище)

Термін подання кваліфікаційної роботи до ЕК 12 червня 2023 року

РЕФЕРАТ

Дипломна робота складається зі вступу, трьох розділів, загальних висновків, списку використаних джерел, додатків, має 92 сторінки основного тексту,1 таблиця, та 10 рисунків. Список використаних джерел містить 30 найменування і займає 3 сторінки.

Об'єктом дослідження є процес виявлення вразливостей в додатках з використанням технології блокчейн.

Предметом дослідження є методи запобігання експлуатації вразливостей віртуальної машини Ethereum.

Методи дослідження використанні при підготовці дипломної роботи:

- аналіз наукової літератури;
- аналіз міжнародних стандартів;
- узагальнення державної та міжнародної практики;
- порівняння та синтез.

В роботі проаналізована існуюча література з безпеки блокчейн додатків, виконаний аналіз документів, порівняння, вивчення та узагальнення вітчизняної і зарубіжної практики з теми блокчейн – технологій, розроблено рекомендації щодо підвищення рівня захисту додатків з використанням технології блокчейн.

Результати досліджень можуть застосовуватися в області інформаційної безпеки, які дають можливість розробникам і користувачам обрати ефективний, найбільш вдалий та дієвий спосіб захисту інформації, яка циркулює в інформаційно-телекомунікаційних системах.

Практична цінність отриманих результатів полягає в розробці рекомендацій з підвищення рівню захисту блокчейн додатків.

Напрямки подальших досліджень: розробка засобів аналізу вразливостей в додатках з використанням технології блокчейн.

Ключові слова: розумні контракти, атака, Ethereum, EVM, методи захисту.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

ETH	–	Ether
DAO	–	Decentralized autonomous organization
PoW	–	Proof of Work
PoS	–	Proof of Stake
DPoS	–	Delegated Proof of Stake
BFT	–	Byzantine Fault Tolerance
EVM	–	Ethereum Virtual Machine
EOA	–	External Owned Adress
API	–	Applied programming interface
DeFi	–	Decentralized Finance
DEX	–	Decentralized Exchange
AMM	–	Automated market maker
EIP	–	Ethereum Improvement Proposal
RBAC	–	Role Based Access Control
DOS	–	Denial of Sevice
UTXO	–	Unspent Transaction Output
KYC	–	Know Your Customer

ЗМІСТ

ВСТУП

Актуальність роботи зумовлюється тим, що додатки з використанням технології блокчейн стають все більш популярними і широко використовуються в різних галузях, таких як фінанси, логістика, медицина, громадська безпека та інші. Однак, разом зі зростанням використання блокчейн додатків з'являються нові виклики та загрози їх безпеці.

Завдяки широкому використанню блокчейну компанії будь-якого розміру і мільйони користувачів по всьому світу отримали доступ до розмаїття фінансових сервісів і нові можливості для побудови децентралізованих додатків. З іншого боку, однією з найбільших проблем блокчейн технології є безпека, оскільки вона базується на розподіленій мережі, де вся інформація зберігається в блоках і підписується цифровими підписами. Проте, існує кілька широковживаних атак, які можуть погрожувати цілісності та безпеці блокчейн додатків, такі як подвійні витрати, 51% атака, атака бічних ланцюгів, розкриття конфіденційної інформації та багато інших.

Для забезпечення безпеки блокчейн додатків існує потреба у розробці та вдосконаленні різних засобів захисту.

Метою роботи є вирішення проблем безпеки, що виникають при використанні блокчейн технології, та розробка рекомендацій з підвищення рівня захищеності сучасних блокчейн додатків.

Для досягнення поставленої мети вирішувалися наступні завдання:

- Проведено аналіз інцидентів повязаних з компроментацією блокчейн технологій;
- Досліджено типові вразливості віртуальної машини Ethereum та оцінено ризики її експлуатації;
- Розроблено модель загроз для сучасних блокчейн додатків;
- Розроблено рекомендацій щодо підвищення рівня захищеності блокчейн додатків**;**

Об’єктом дослідження є процес виявлення вразливостей в додатках з використанням технології блокчейн.

Предметом дослідження є методи запобігання експлуатації вразливостей віртуальної машини Ethereum.

Методи дослідження використанні при підготовці дипломної роботи:

- аналіз наукової літератури;
- аналіз міжнародних стандартів;
- узагальнення державної та міжнародної практики;
- порівняння та синтез.

РОЗДІЛ 1
АНАЛІЗ ТЕХНОЛОГІЇ БЛОКЧЕЙН

1.1 Дослідження технології блокчейн

Блокчейн нещодавно з'явився як дослідницький тренд з потенційним застосуванням у широкому спектрі галузей та контекстів. Однією з найбільш успішних технологій блокчейну є смарт-контракти, які широко використовуються в комерційному середовищі (наприклад, у фінансових операціях на великі суми). Це, однак, має наслідки для безпеки через можливість отримати фінансову вигоду від інциденту безпеки (наприклад, виявлення та використання вразливості в смарт-контракті або його реалізації). Серед усіх блокчейнів Ethereum є найбільш активним і вражаючим. В даній роботі було розглянути як смарт-контракти можуть бути використані зловмисниками та стати мішенню для зловмисників, зокрема, на питаннях безпеки контрактної моделі програми, вразливостях в програмі та питаннях безпеки, що вносяться середовищем виконання програми. Чому безпека смарт-контрактів є важливою? По-перше, такі контракти, як правило, використовуються у фінансовій сфері, а отже, вони є привабливою мішенню для фінансово та кримінально мотивованих кіберзлочинців. Крім того, будь-який успішний злам, особливо той, що набув широкого розголосу, може вплинути на віру спільноти в смарт-контракти, а отже, і на їх використання. За останні роки відбулася невелика кількість таких інцидентів, наприклад, інциденти, пов'язані з подією Децентралізованої автономної організації (DAO) і Parity. Зовсім недавно, у 2018 році, зловмисники зламали Fomo 3D, ігровий Dapp, провівши фронт-запуск і ввівши в мережу Blockchain багато транзакцій з високими комісіями, щоб запобігти пакуванню інших транзакцій, в результаті чого було викрадено близько 10469,66 ETH.

Технологія блокчейн, незмінний, децентралізований публічний реєстр, з моменту свого створення зробила революцію в обміні інформацією в багатьох галузях. У даному розділі досліджуються фундаментальні механізми, які регулюють роботу технології блокчейн, і розглядаються її значні наслідки в різних секторах.

Технологія блокчейн, вперше впроваджена у 2009 році як основний компонент біткоїна, першої децентралізованої криптовалюти, зараз перетворилася на технологію, що здатна суттєво змінити різні аспекти життя суспільства - від фінансів та управління ланцюгами поставок до систем голосування та верифікації особистості. Цей звіт має на меті висвітлити фундаментальні механізми, що лежать в основі технології блокчейн, а також висвітлити широкий спектр застосувань і майбутніх можливостей, які вона пропонує.

Блокчейн - це, по суті, ланцюжок блоків, де кожен блок містить список транзакцій. Блоки пов'язані один з одним за допомогою криптографічного механізму, створюючи ланцюжок блоків - звідси і термін "блокчейн". Його можна вважати публічною розподіленою книгою, доступною лише для додавання, тобто після того, як дані записані, вони не можуть бути змінені без зміни всіх наступних блоків і отримання консенсусу в мережі. Кожен блок в блокчейні зазвичай складається з Блок-даних: Фактичних деталей транзакцій, які залежать від типу блокчейну. Наприклад, блок Bitcoin містить дані про відправника, одержувача та кількість монет [1].

Нонсе: Nonce ("номер, який використовується тільки один раз") - це номер, доданий до хешованого або зашифрованого блоку в блокчейні, який при повторному хешуванні відповідає обмеженням рівня складності. Nonce - це число, яке шукають майнери блокчейну.

Хеш: Унікальний ідентифікатор блоку, згенерований за допомогою хеш-алгоритму (наприклад, SHA-256 в блокчейні Bitcoin). Хеш однозначно представляє дані всередині блоку.

Хеш попереднього блоку: Хеш попереднього блоку в ланцюжку. Це створює ланцюжок блоків і робить блокчейн безпечним і незмінним.

Кожна транзакція в блоці - це запис, який був підтверджений учасниками мережі блокчейн. Коли транзакцію здійснено, вона транслюється на всі вузли мережі. Потім вузли використовують публічний ключ відправника для перевірки підпису на транзакції, забезпечуючи її автентичність і те, що вона не була підроблена. Після підтвердження транзакція потрапляє в пул непідтверджених транзакцій.

Хешування - це криптографічний метод, який перетворює вхідні дані будь-якої довжини в рядок тексту фіксованого розміру. Унікальне хеш-значення відіграє вирішальну роль у підтримці цілісності блокчейну. Будь-яка зміна вмісту блоку змінює його хеш, попереджаючи систему про можливе втручання.

На відміну від традиційних централізованих баз даних, блокчейн використовує розподілений реєстр, де кожен учасник або вузол мережі зберігає копію всього блокчейну. Така децентралізація сприяє стійкості та безпеці системи, роблячи її практично несприйнятливою до одномоментних збоїв або атак.

Основні процеси в операціях блокчейну включають перевірку транзакцій, створення блоків і механізми консенсусу.

Коли транзакція ініціюється, вона повинна бути перевірена вузлами мережі. Це передбачає перевірку дійсності транзакції на основі набору заздалегідь визначених правил і попередніх транзакцій у блокчейні.

Після перевірки транзакції об'єднуються в новий блок. Цей блок додається до блокчейну, стаючи постійною, незмінною частиною реєстру.

Процес додавання нового блоку до блокчейну регулюється механізмами консенсусу. Механізми консенсусу - це протоколи, які забезпечують синхронізацію всіх вузлів між собою і погоджують, які блоки є дійсними і можуть бути додані до блокчейну.

Доказ роботи (Proof of Work, PoW): Перший механізм консенсусу в блокчейні, який використовується Біткоін. Він вимагає від користувача довести, що певна робота була виконана у вигляді розв'язання складної обчислювальної головоломки.

Proof of Stake (PoS): Цей тип алгоритму консенсусу передбачає вибір творця нового блоку на основі його частки або багатства в мережі. Це може бути засновано на кількості токенів, які людина тримає або "вклала" в якості застави.

Делегований доказ частки (DPoS): У DPoS зацікавлені сторони делегують свої повноваження кільком представникам, які підтверджують транзакції і створюють блоки від їх імені.

Візантійська відмовостійкість (BFT): У механізмах консенсусу в стилі BFT певні вузли (відомі як валідатори) відповідають за перевірку транзакцій і додавання їх до блокчейну. BFT може витримати до третини зловмисних дій з боку вузлів.

Кожен механізм консенсусу має свої сильні і слабкі сторони з точки зору масштабованості, безпеки і децентралізації, і різні блокчейни використовують різні механізми консенсусу в залежності від своїх конкретних потреб і цілей. Прозорість, незмінність і децентралізація, пропоновані технологією блокчейн, відкривають двері для безлічі додатків за межами криптовалюти. Це, зокрема, смарт-контракти, відстеження ланцюгів поставок, децентралізовані фінанси, управління медичною документацією та безпечні системи голосування.

Технологія блокчейн - це проривна сила, яка здатна трансформувати багато секторів, пропонуючи підвищену безпеку, прозорість та ефективність. Розуміння того, як вона працює, може призвести до кращого використання та подальшого розвитку, сприяючи майбутньому, де довіра встановлюється не центральними органами, а консенсусом і криптографією.

1.2 Дослідження технології Ethereum

Ethereum, розроблений Віталіком Бутеріним та іншими і випущений у 2015 році, являє собою значне розширення та еволюцію принципів і корисності технології блокчейн, вперше втіленої в біткоїні. У той час як біткойн був створений як децентралізована цифрова валюта, Ефіріум був розроблений як децентралізована платформа, яка дозволяє розробникам створювати і розгортати смарт-контракти і децентралізовані додатки (dApps).

Віртуальна машина Ethereum (EVM) є ключовим компонентом мережі Ethereum. Це частина протоколу, яка фактично виконує смарт-контракти.

EVM - це, по суті, глобальний децентралізований комп'ютер, що містить мільйони виконуваних додатків, які працюють на мережі з тисяч комп'ютерів. Його основна функція полягає в оновленні стану блокчейну Ethereum шляхом виконання серії операцій, відомих як опкоди, які складають смарт-контракти [2].

Смарт-контракти - це сценарії спеціальної бізнес-логіки, написані мовою високого рівня, наприклад, Solidity, які існують за певною адресою в блокчейні Ethereum, але не мають приватного ключа. Після розміщення коду смарт-контракту в блокчейні його неможливо змінити, що робить виконання та результати контракту незмінними. Коли за адресою контракту здійснюється транзакція, EVM виконує код контракту. Залежно від складності контракту, це може включати читання або запис даних, обчислення, виклик інших контрактів або навіть укладання нових контрактів.

Кожна операція, яку виконує EVM, вимагає певної кількості "газу". Газ - це міра обчислювальної роботи в мережі Ethereum, яка оплачується в ефірі. Система газу призначена для запобігання спаму в мережі і пропорційного розподілу ресурсів.

EVM також повністю ізольований від мережі, файлової системи та інших процесів хост-комп'ютера. Така "пісочниця" гарантує, що смарт-контракти не можуть впливати на зовнішні системи, пропонуючи високий рівень безпеки. Єдиний спосіб взаємодії зі смарт-контрактом - це транзакція в мережі Ethereum.

Однією з ключових особливостей EVM є те, що він є "повним за Тьюрінгом", тобто може виконати будь-який алгоритм, маючи достатньо ресурсів. Це критична відмінність від біткоїна, який використовує для своїх скриптів мову, що не є повною за Тюрінгом, для додаткової безпеки.Ethereum використовує режим рахунку, подібний до механізму управління рахунками у традиційній банківській системі. Існує два типи рахунків, а саме: зовнішні (EOA) та контрактні. Обидва типи облікових записів унікально ідентифікуються 20-байтною адресою (тобто, їх ідентифікаторами в мережі Ethereum). EOA

контролюються парами публічних/приватних ключів, в основному використовуються для управління ефірами і взаємодії з контрактами шляхом надсилання транзакцій. В той час як контрактні рахунки контролюються кодами без ключів і в основному використовуються для реалізації різноманітних функціональних вимог і запису змін стану контрактів, таких як виконані транзакції і модифікація балансу. На відміну від EOA, контрактні рахунки не можуть надсилати транзакції, але вони можуть надсилати повідомлення для виклику інших контрактів [3]. Крім того, контрактні акаунти не можуть проактивно взаємодіяти з EOA, але вони можуть використовувати деякі "радикальні" механізми, такі як самознищення (коли всі ефіри, які вони утримують, будуть повернуті їхнім творцям після успішного виконання). У контрактному акаунті також є місце для зберігання хешу коду, який можна використовувати для пошуку кодів, тоді як EOA не зберігає цю цінність. З точки зору схожості, обидва типи акаунтів зберігають nonce, баланс активів і кореневий хеш всіх збережених станів. Nonce - це механізм, розроблений для захисту від атак у відповідь і подвійних витрат, який буде збільшуватися на 1, як тільки акаунт відправляє транзакцію. Іншими словами, якщо хтось транслює транзакцію знову і знову, сподіваючись, що вона буде виконана майнерами багато разів, майнери ідентифікують її як повторну транзакцію (завдяки nonce) і відкинуть її. Очевидно, що якщо зловмисник спробує використати той самий nonce, щоб полегшити подвійні витрати, то ціни на газ, додані до транзакції, визначать переможця, оскільки вони, швидше за все, будуть оброблені першими.Транзакції та взаємодія Транзакції будуть транслюватися кожному майнеру для виконання і змінюватимуть стан сховища Blockchain після досягнення консенсусу. У кожній транзакції буде вказано номер облікового запису (як описано вище), ціну за газ, максимальний платіж за газ для цієї конкретної транзакції, передану вартість, одержувача, вхідні дані та підпис. З усієї інформації, що міститься в ньому, механізм нарахування плати за газ відіграє ключову роль у багатьох аспектах, наприклад: 1) компенсація і стимул для майнерів, які виконують і зберігають транзакції; 2) контрзахід для запобігання атакам на відмову в обслуговуванні

(DoS), наприклад, зловмисникам, які надсилають складні для обчислення транзакції. Транзакція з нульовим одержувачем буде розглядатися як запит на створення контракту. В такому випадку майнери виконають байткоди, що входять до складу корисного навантаження. Тому корисне навантаження включає не тільки коди контрактів, але й початкові коди та параметри в конструкторі. Початкові коди відповідають за створення облікового запису контракту, зберігання кодів, ініціалізацію функцій конструктора тощо. Потім вони будуть відкинуті після завершення виконання. коли одержувачем є адреса контракту, майнери будуть викликати відповідний контракт. функція "Call", яка також є своєрідним методом для виклику контрактів, із зовнішнім API Параметри контракту Функція виклику контракту, що лежить в основі API. На відміну від транзакцій, "Виклик" не буде поширюватися в мережі або отримувати консенсус, і вони є локальними викликами шляхом читання локальної бази даних стану без внесення будь-яких змін, придатними для перегляду, чистими і постійними функціями в Solidity. Хоча вони не беруть плату, вони вийдуть з ладу, якщо не вистачатиме газу. Це тому, що вони повинні бути виконані в місцевому EVM. Контракти можуть бути викликані як EOA, так і іншими акаунтами за допомогою функції "Виклик". Очевидно, що вони можуть бути викликані з EOA за допомогою транзакцій. Як вони можуть бути викликані контрактами за допомогою транзакцій? Ну, контракти можуть взаємодіяти один з одним, і вони використовують "виклик повідомлення", а не транзакції. Теоретично кажучи, "виклик повідомлення" - це також різновид транзакції, який дозволяє абонентам швидко отримувати значення, що повертаються.EVM та виконання EVM - це архітектура, що базується на стеку, а не на регістрі, і забезпечує відокремлене середовище виконання, щоб захистити виконання контрактів від зовнішніх атак та уникнути впливу зловмисних контрактів на всю систему. Існує три типи структури зберігання даних для полегшення роботи, а саме: стек, пам'ять і сховище. Стек можна використовувати для безкоштовного зберігання локальних змінних, а пам'ять використовується для зберігання параметрів і значень, що повертаються, які будуть звільнені після виклику функцій. Стек і пам'ять є

мінливими, вони очищаються після завершення транзакції. Сховище буде зберігати змінні стану постійно, тому воно є дорогим. Після отримання викликів до контрактів (через повідомлення або транзакції), EVM спочатку шукає і завантажує код контракту з локальної бази даних. На відміну від звичайного програмного забезпечення, яке має лише одну точку входу (main()), всі публічні функції в смарт-контрактах можуть бути точками входу. Тому перші чотири байти корисного навантаження, сигнатура функції, вказують на функцію виклику, за якими йдуть параметри функції. EVM знайде відповідні байткоди функції і розбере їх на опкоди байт за байтом. Більше того, кожен опкод пов'язаний з детальним описом операції. На основі цих інструкцій EVM буде обробляти ці параметри. Машина може генерувати виключення з таких причин, як переповнення стеку або невірні інструкції. Таким чином, вразливості контракту в кінцевому підсумку будуть використані в EVM. Результати виконання (включаючи зміну балансу учасників, журнали виконання і квитанції) будуть упаковані як "StorageObject", що зберігається в книзі блокчейну. Оскільки весь цей процес повторюється всіма вузлами в системі Blockchain, до тих пір, поки скомпрометованих вузлів менше, ніж більшість, Blockchain все ще може працювати стабільно. Компрометація декількох вузлів не вплине на загальну безпеку системи. Однак ми повинні гарантувати, що один і той же смарт-контракт на всіх вузлах отримує однакові транзакції, дані і генерує ідентичні результати.

Загальну схему роботи розумного контракту Ethereum проілюстровано на рисунку 1.1:

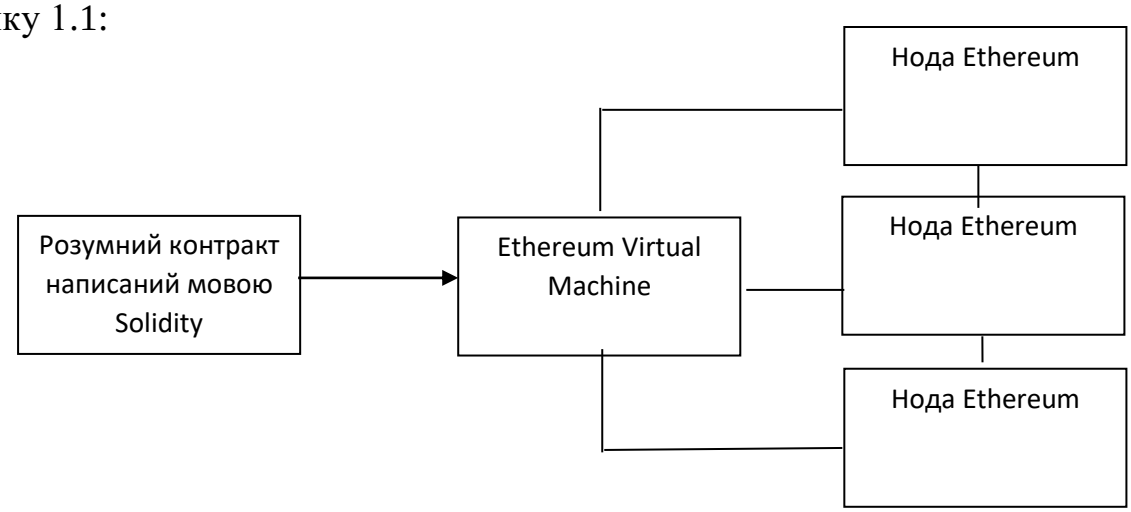

Рисунок 1.1 - Схема роботи віртуальної машини Ethereum.

1.3 Види додатків на блокчейні Ethereum

Смарт-контракти - це самодостатні контракти, умови яких безпосередньо записані в коді. У блокчейні Ethereum ці контракти зберігаються у вигляді байт-коду за певною адресою, а їх виконання запускається транзакціями або повідомленнями, отриманими від інших контрактів. Смарт-контракти можуть моделювати складні фінансові взаємодії, мати кілька підписантів, керувати угодами між користувачами, надавати корисність іншим контрактам, зберігати інформацію про програму або, по суті, функціонувати як "мультипідписні" облікові записи [4]. Функціональність смарт-контрактів у блокчейні Ethereum практично безмежна, але з'явилося кілька стандартних категорій:

1.3.1 Фінансові контракти

Фінансові контракти часто є основою протоколів DeFi (децентралізованих фінансів). Вони можуть бути запрограмовані для автоматизації виконання фінансових угод, таких як кредитування і запозичення, процентні свопи, страхові контракти, ф'ючерсні та опціонні контракти.

Фінансові контракти, які є різновидом смарт-контрактів в мережі Ethereum, втілюють складні фінансові операції в децентралізованому середовищі. Ці фінансові операції охоплюють широкий спектр діяльності, включаючи, але не обмежуючись, кредитування і запозичення, процентні свопи, страхові контракти і торгівлю деривативами.

Ось деякі з конкретних типів фінансових контрактів:

- Договори позики та кредитування:

Це один з найпопулярніших типів фінансових контрактів. Вони створюють децентралізовані платформи, де користувачі можуть позичати і позичати кошти безпосередньо між собою, усуваючи потребу в традиційному фінансовому посереднику.

- Типові функції: lend, borrow, repay, withdraw, deposit, liquidate.

- Ключові слова public, payable, view, returns, require.

Наприклад, на протоколах Compound або Aave, коли користувач хоче позичити активи, він взаємодіє з кредитним смарт-контрактом протоколу, надсилаючи транзакцію, яка викликає функцію "lend" або "deposit". Ця функція оновлює баланс кредитора і загальну пропозицію активу на платформі.

- Контракти зі стейблкоінами:

Стейблкоіни - це криптовалюти, які прив'язують свою вартість до резерву активів, часто до фіатних валют, таких як долар США. Основою стейблкоінів є смарт-контракти, які керують механізмами випуску, погашення та стабільності.

- Типові функції: mint, burn, transfer, balanceOf,.

- Ключові слова: public, view, returns.

Наприклад, DAI, стейблкоїн, випущений за протоколом MakerDAO, використовує систему смарт-контрактів для забезпечення підтримки вартості, приблизно еквівалентної долару США.

- Контракти на фармінгу прибутку:

Фармінг передбачає використання різних протоколів DeFi для отримання прибутку або винагороди. Це досягається за допомогою смарт-контрактів, які керують взаємодією між протоколами і автоматично оптимізують для отримання найкращого прибутку.

- Типові функції: stake, unstake, claimRewards.

- Ключові слова: public, view, returns, require.

Такий протокол, як Yearn.Finance, використовує складні смарт-контракти, які переміщують кошти між різними протоколами DeFi, щоб оптимізувати прибутковість для своїх користувачів.

- Деривативні контракти:

Деривативи - це фінансові контракти, які отримують свою вартість від базового активу. У DeFi деривативи можуть бути закодовані в смарт-контракти, що дозволяє створювати синтетичні активи, які відстежують ціну реальних активів.

- Типові функції: mint, burn, collateralize, redeem.

- Ключові слова: public, payable, view, returns.

Synthetix - приклад протоколу, який використовує смарт-контракти для створення синтетичних активів, що відстежують ціну активів, таких як криптовалюти, товари, акції та індекси.

- Страхові контракти:

Страхові контракти використовують смарт-контракти для захисту від помилок смарт-контрактів, злому або інших ризиків DeFi. Користувачі платять внески, щоб отримати страховку, а претензії обробляються і виплачуються через смарт-контракт.

- Типові функції: buyCoverage, submitClaim, voteOnClaim тощо.

- Ключові слова: public, payable, view, returns.

Nexus Mutual - це протокол, який забезпечує покриття смарт-контрактів, де весь життєвий цикл страхового полісу обробляється смарт-контрактом.

Важливо відзначити, що завдяки відкритій та інтероперабельній природі Ethereum, багато протоколів DeFi використовують кілька типів фінансових контрактів, створюючи взаємопов'язану та автоматизовану фінансову систему. Оскільки цей простір продовжує розвиватися та впроваджувати інновації, ймовірно, з'являться нові типи фінансових контрактів.

1.3.2 Контракти токенів

Контракти токенів, також відомі як стандарти токенів, - це набір правил, які визначають, як можна передавати токени, як затверджувати транзакції, як користувачі можуть отримати доступ до даних про токени тощо. Вони схожі на план, який гарантує, що всі токени (як взаємозамінні, так і не взаємозамінні) поводяться передбачувано і розпізнаються різноманітними гаманцями, біржами та іншими смарт-контрактами, які з ними взаємодіють. Ось три найпоширеніші токен-контракти на Ethereum:

Токени ERC-20

ERC-20 (Ethereum Request for Comments 20) - це стандартний інтерфейс для взаємозамінних токенів, тобто всі токени ідентичні один одному і взаємозамінні. Це найпоширеніший тип токенів на Ethereum.

- Типові функції:

- переказ: Переміщує вказану кількість токенів з облікового запису відправника повідомлення на інший обліковий запис.
- затвердити: Схвалює переказ певної кількості токенів з акаунта відправника повідомлення іншим суб'єктом.
- allowance: Повертає кількість токенів, яку витрачальник може переказати від імені власника.
- balanceOf: Повертає баланс токенів будь-якого облікового запису Ethereum.
- totalSupply: Повертає загальний запас токенів.

Приклад: Багато токенів, що використовуються в DeFi, такі як UNI (Uniswap), LINK (Chainlink) або COMP (Compound), є токенами ERC-20. Ці токени можуть представляти широкий спектр цифрових активів, таких як валюти, товари, процентні інструменти, права на управління тощо.

Токени ERC-721

ERC-721 - це стандарт для представлення не взаємозамінних токенів (NFT) - унікальних токенів з індивідуальними характеристиками, які відрізняють їх один від одного. Цей стандарт дозволяє представляти право власності на унікальні предмети або активи.

- Типові функції:

- balanceOf: Повертає кількість NFT, що належать обліковому запису Ethereum.
- ownerOf: Повертає власника конкретного NFT.
- safeTransferFrom: Безпечно передає право власності на конкретний NFT з одного облікового запису на інший.
- transferFrom: Передає право власності на конкретний NFT з одного акаунта на інший.

- approve: Затверджує іншу адресу для передачі конкретного NFT.
- getApproved: Повертає затверджену адресу для конкретного NFT.
- setApprovalForAll: затверджує або видаляє дозвіл третій стороні ("оператору") на управління всіма токенами відправника повідомлення.

Приклад: CryptoKitties, віртуальна гра на основі блокчейну, яка дозволяє гравцям всиновлювати, вирощувати і торгувати віртуальними котиками, є одним з перших і найбільш відомих прикладів використання токенів ERC-721.

ERC-1155 Tokens

ERC-1155 - це стандарт токенів, який може представляти будь-яку кількість взаємозамінних і не взаємозамінних елементів в рамках одного контракту. Він поєднує в собі можливості ERC-20 і ERC-721 і призначений для більш ефективної роботи з токенами.

- Типові функції:

- balanceOf: Повертає баланс токенів на даному рахунку.
- balanceOfBatch: Повертає баланс декількох токенів, що належать декільком акаунтам.
- setApprovalForAll: Встановлює або скасовує дозвіл третьої сторони ("оператора") на управління всіма токенами відправника повідомлення.
- isApprovedForAll: Повертає, чи дозволено оператору управляти всіма токенами власника.
- safeTransferFrom: Безпечно переказує токени з одного акаунта на інший.
- safeBatchTransferFrom: Безпечно переказує кілька типів токенів з одного акаунта на інший.

Приклад: Enjin, проект, що створює набір ігрових продуктів на основі блокчейну, використовує ERC-1155 для випуску взаємозамінних (для віртуальних валют) і не взаємозамінних токенів (для унікальних цифрових активів) для використання в іграх.

1.3.3. Децентралізовані автономні організації (DAO)

Децентралізовані автономні організації, або ДАО, представляють собою зміну парадигми в тому, як ми концептуалізуємо і структуруємо організації. ДАО - це, по суті, онлайн-цифрові організації, які функціонують за допомогою впровадження попередньо закодованих правил. Ці правила застосовуються на блокчейні, що усуває потребу в централізованому управлінні.

Оскільки DAO розгорнуті на блокчейні, вони є відкритими та прозорими, а правила, за якими вони працюють, є незмінними, якщо тільки вони не змінені консенсусом. DAO зазвичай мають власний токен, а власники токенів часто мають право голосу, пропорційне до їхньої частки. Ці права голосу можуть бути використані для прийняття рішень про зміни в протоколі, розподіл казначейських коштів тощо [5].

DAO мають безліч потенційних варіантів використання. Вони можуть використовуватися для децентралізованого управління протоколом, наприклад, Uniswap або Compound, де власники токенів можуть голосувати за зміни параметрів протоколу або за розподіл коштів зі скарбниці протоколу.

DAO також можуть бути сформовані навколо спільних інвестиційних цілей. Прикладом цього є DAO Moloch, де учасники об'єднують кошти і голосують за інвестиційні рішення. Коли робиться пропозиція інвестувати в проект, учасники можуть проголосувати "за", "проти" або вирішити вийти з DAO і забрати свою частку об'єднаних коштів з собою.

Інший варіант використання - створення DAO для децентралізованого створення або курації контенту, як у випадку з DAOstack, який передбачає майбутнє "децентралізованих компаній", де широка мережа людей співпрацює для досягнення спільних цілей і отримує винагороду за допомогою токенів DAO.

DAO також використовуються для створення децентралізованих версій традиційних організаційних структур. Наприклад, Aragon надає платформу для створення та управління DAO з широкими можливостями налаштування, які

дозволяють створювати децентралізовані версії компаній, некомерційних організацій, клубів тощо.

Типові функції DAO

- голосування: Дозволяє учасникам голосувати за пропозицію. Голосування може бути за пропозицію або проти неї.

- делегувати: Дозволяє члену делегувати право голосу іншому члену.

- запропонувати: Дозволяє члену запропонувати зміни до DAO. Конкретні вимоги для внесення пропозиції можуть відрізнятися, і часто вимагається мінімальний баланс токенів або застава.

- executeProposal: Якщо пропозиція прийнята, ця функція викликається для реалізації пропозиції. У деяких DAO це відбувається автоматично, а в інших учасник повинен викликати цю функцію.

- внести або зняти стейк: у деяких ДАО учасники повинні внести токени для участі в управлінні або для отримання винагороди. Ці функції дозволяють вносити та знімати стейки.

1.3.4. Децентралізовані біржі (DEX)

Децентралізовані біржі (DEX) - це криптовалютні біржі, які працюють без центрального органу, що дозволяє здійснювати однорангову торгівлю безпосередньо між користувачами. Така децентралізована природа може забезпечити ряд переваг у порівнянні з традиційними централізованими біржами, включаючи підвищену конфіденційність, меншу залежність від третьої сторони і часто нижчі комісії. Функціональність децентралізованих бірж (DEX) забезпечується смарт-контрактамиОсновна функція DEX - полегшити торгівлю криптовалютами. Це досягається за допомогою смарт-контрактів, які забезпечують дотримання правил торгівлі, узгодження замовлень і переказ коштів. Залежно від архітектури DEX, угоди здійснюються або безпосередньо з гаманця на гаманець (як у випадку з Uniswap), або через систему проксі-серверів смарт-

контрактів (як у випадку з 0x). Давайте розглянемо докладніше, як ці контракти функціонують в рамках DEX.

Контракти автоматизованих маркет-мейкерів (AMM)

Автоматизовані маркет-мейкери (AMM), такі як Uniswap або SushiSwap, використовують тип смарт-контракту, який створює пул ліквідності з двох токенів. Коли користувачі взаємодіють з контрактом, вони можуть торгувати токенами безпосередньо з пулу, надавати пулу ліквідність для отримання комісійних або вилучати свою ліквідність.

Ось ключові функції в AMM-контракті:

- обмін точних токенів на токени: Дозволяє користувачеві обміняти певну кількість одних токенів на інші.

- addLiquidity: Дозволяє користувачеві додати рівну кількість двох токенів до пулу ліквідності. Натомість він отримує токени ліквідності, які представляють його частку в пулі.

- removeLiquidity: Дозволяє користувачеві спалити свої токени ліквідності, щоб видалити свою частку ліквідності з пулу.

DEX-контракти книги ордерів

DEX книги ордерів, такі як 0x, використовують інший тип контракту. Замість пулу токенів вони мають книгу замовлень, де користувачі можуть розміщувати замовлення на купівлю або продаж. Ці замовлення потім можуть бути виконані іншими користувачами.

Основні функції в DEX-контракті з книгою замовлень включають в себе

- createOrder: За допомогою цієї функції користувач може створити нове замовлення, вказавши токени, якими він хоче торгувати, кількість і ціну.

- cancelOrder: Ця функція дозволяє користувачеві скасувати відкритий ордер, який він розмістив.

- fillOrder: Інший користувач може заповнити відкритий ордер за допомогою цієї функції, здійснивши торгівлю.

Спільні функції DEX

Деякі функції є спільними для обох типів DEX, зокрема

- transferFrom: Ця функція використовується для передачі токенів від одного користувача до іншого. В контексті DEX вона використовується для переміщення токенів від покупця до продавця (і навпаки) під час торгівлі.

- затвердити: Перш ніж токени користувача можуть бути переміщені за контрактом DEX, він повинен схвалити контракт на переміщення певної кількості своїх токенів. Ця функція викликається для надання такого схвалення.

1.3.5. Оракули

Оракули - це смарт-контракти, які забезпечують зв'язок між блокчейном і зовнішнім світом, дозволяючи іншим смарт-контрактам запускати дії на основі реальних даних.

Chainlink - це децентралізована мережа оракулів, яка дозволяє смарт-контрактам безпечно взаємодіяти з реальними даними і сервісами за межами їхніх власних блокчейнів.

Важливо відзначити, що це лише деякі з типів і способів використання смарт-контрактів, а повний спектр можливостей, яких можна досягти з їх допомогою, величезний і постійно розширюється.

Оракули є важливим компонентом для смарт-контрактів і блокчейн-додатків, які потребують реальних даних або взаємодіють із зовнішніми системами. Оскільки блокчейн є детермінованою, автономною системою, він не може отримати доступ до даних або взаємодіяти з ними за межами своєї мережі. Оракули - це міст, який з'єднує внутрішній світ блокчейну і смарт-контрактів з зовнішнім світом, надаючи необхідні дані для виконання смарт-контрактів, які покладаються на конкретні умови реального світу.

Оракули можна класифікувати на основі декількох різних факторів, включаючи тип даних, які вони надають, джерело даних, напрямок даних і рівень децентралізації.

1. Оракули типу даних: Ці оракули можна розділити на кілька типів: цінові оракули, які надають інформацію про ціни на активи; погодні оракули, які

надають інформацію про погоду; і оракули подій, які надають дані про результати конкретної події.

2. Оракули за типом джерела: Апаратні оракули збирають дані з фізичного світу (наприклад, з пристроїв Інтернету речей), а програмні оракули збирають дані з онлайн-джерел.

3. Напрямок оракулів даних: Ці оракули можуть бути вхідними (надавати дані для смарт-контрактів) або вихідними (відправляти дані з блокчейну в зовнішній світ).

4. Централізація оракулів: Централізовані оракули контролюються однією організацією, в той час як децентралізовані оракули контролюються декількома організаціями для підвищення довіри і зменшення центральних точок відмови.

Ключові функції в смарт-контрактах Oracle

Типові функції в смарт-контрактах Oracle можуть сильно відрізнятися в залежності від варіанту використання та дизайну, але деякі з них є загальними:

- функція запиту: Викликається користувачем або смарт-контрактом для запиту даних.

- функція зворотного виклику: Після отримання даних, оракул викликає цю функцію, щоб відправити дані назад до контракту, який їх запитував.

- функція оновлення: У деяких випадках оракули постійно оновлюють певні дані (наприклад, ціну активу) самостійно. Ця функція використовується для виконання оновлення.

Варіанти використання та приклади оракулів

Оракули використовуються в багатьох децентралізованих додатках (dApps) і платформах, де зовнішня інформація необхідна для виконання контракту. Наприклад, ринки прогнозування, такі як Augur, страхові додатки, такі як Etherisc, і стейблкоіни, такі як Dai, потребують реальних даних, що надаються оракулами.

У DeFi оракули широко використовуються для надання цінових даних. Такі проекти, як Chainlink, Band Protocol та API3, є популярними рішеннями для

оракулів у просторі DeFi, які отримують цінові дані з декількох зовнішніх API та безпечно і децентралізовано передають їх у мережеві протоколи.

Висновки за розділом 1

Технологія блокчейн - це децентралізована система розподіленого реєстру, яка зберігає дані в декількох системах по всьому світу, щоб підвищити прозорість і запобігти зміні даних. Кожен блок даних пов'язаний з попереднім і наступним, що гарантує безпеку та незмінність інформації. Блокчейн управляється одноранговими мережами і захищений за допомогою криптографічних принципів.

Архітектура блокчейну складається з трьох основних компонентів:

Блок: Кожен блок містить набір транзакцій. Він складається із заголовка (що включає метадані, такі як унікальний ідентифікаційний номер блоку, час створення блоку та посилання на попередній блок) та секції вмісту (з детальним описом транзакцій).

Ланцюжок: Ланцюжок з'єднує всі блоки. Кожен наступний блок в ланцюжку містить криптографічний хеш попереднього блоку, утворюючи пов'язаний ланцюжок, в якому дані не можуть бути змінені без впливу на весь ланцюжок.

Мережа: Блокчейн використовує децентралізовану мережу, де кожен учасник (вузол) отримує копію всього ланцюжка. Ноди перевіряють нові блоки і відповідають за узгодження стану кожного блоку за допомогою алгоритмів консенсусу.

Ethereum - це заснована на блокчейні платформа з відкритим вихідним кодом для децентралізованих додатків. Вона використовує власну криптовалюту Ефір (ETH) для здійснення транзакцій в мережі. Ключовою особливістю Ethereum є підтримка смарт-контрактів, які є самодостатніми контрактами, умови яких безпосередньо прописані в коді.

В Ethereum існує два типи смарт-контрактів:

Облікові записи, що належать ззовні (EOA): Ці акаунти контролюються приватними ключами і не мають пов'язаного з ними коду. Вони можуть

взаємодіяти з іншими акаунтами, створюючи транзакції, закодовані з потрібною їм функціональністю.

Контрактні акаунти: Ці акаунти контролюються за допомогою коду контракту і можуть виконувати операції лише тоді, коли їм це доручено EOA. Вони мають пов'язаний з ними код і сховище, і коли викликається функція з облікового запису контракту, запускається його код, що дозволяє йому читати/записувати внутрішнє сховище, створювати інші контракти або викликати функції з інших контрактів.

Основна мета смарт-контрактів Ethereum полягає в тому, щоб полегшити, перевірити і забезпечити виконання переговорів або виконання контракту в безпечний, прозорий і децентралізований спосіб..

РОЗДІЛ 2
ТЕХНОЛОГІЇ РЕАЛІЗАЦІЇ АТАК НА БЛОКЧЕЙН ДОДАТКИ

2.1 Види атак на блокчейн додатки

Розумні контракти стали фундаментальною основою функціональності технології блокчейн, особливо у сфері децентралізованих фінансів (DeFi). Незважаючи на очевидні переваги, розумні контракти піддаються різним формам атак, що викликає значні занепокоєння з точки зору безпеки. Проаналізуємо значення цих атак, їх механізми та потенційні стратегії пом'якшення наслідків для забезпечення цілісності та безпеки смарт-контрактів.

Вразливості смарт-контрактів часто виникають через помилки в програмуванні, логіку та непередбачувану взаємодію з іншими контрактами. Зокрема, це атаки повторного входу, як показано на прикладі сумнозвісної атаки на DAO у 2016 році, коли рекурсивні виклики використовували контракт для виведення Ефіру на суму близько 60 мільйонів доларів США. Інші типи атак включають в себе атаки на залежність від мітки часу, недоповнення і переповнення, а також атаки на випередження. Ці атаки підкреслюють вразливість смарт-контрактів до маніпуляцій, що призводять до неочікуваних і руйнівних результатів.

Фінансові наслідки атак на смарт-контракти приголомшливі. Властива цим контрактам грошова вартість, що зберігається і передається в рамках цих контрактів, часто робить їх прибутковими цілями для зловмисників. Атаки можуть призвести до значних фінансових втрат, підриваючи стабільність і потенційну прибутковість платформ на основі блокчейну.

Успіх технології блокчейн значною мірою залежить від довіри до її механізмів безпеки. Успішні атаки підривають цю довіру, завдаючи шкоди репутації залучених суб'єктів і потенційно сповільнюючи впровадження технології блокчейн.

На відміну від традиційних цифрових транзакцій, операції на блокчейні, особливо ті, що пов'язані зі смарт-контрактами, є незмінними. Незворотність цих

транзакцій додає додатковий рівень ризику, роблячи успішну атаку дуже шкідливою. Крім того, взаємопов'язаність систем блокчейн посилює наслідки атаки, що може спричинити системні ризики в екосистемі.

Хоча смарт-контракти представляють собою трансформаційний підхід до систем цифрових угод і транзакцій, вони не позбавлені недоліків. Розуміння можливих атак, їхніх наслідків та потенційних заходів протидії має вирішальне значення для їхнього безпечного та ефективного впровадження. Розглянемо найрозповсюдженіші види атак [6].

2.1.1 Атака відтворення

Атака відтворення полягає у відтворенні інформації про транзакцію. Користувач підписує повідомлення, завантажує його в контракт, а потім перевіряє підпис всередині контракту. Але оскільки інформація про підпис користувача знаходиться в мережі, її може отримати будь-хто. При перевірці підпису користувача в контракті, якщо підписане повідомлення не містить змінних, які випадковим чином змінюються з кількістю транзакцій, таких як мітка часу, nonce і т.д., зловмисник буде володіти підписом користувача і підробляти транзакції, отримуючи таким чином прибуток. У широкому розумінні це можна розглядати як процес використання однієї і тієї ж платіжної інформації для купівлі товарів кілька разів. Коли після хардфорку з'явилися ланцюжки Ethereum і Ethereum Classic, було виявлено, що транзакції в ланцюжку Ethereum залишаються дійсними, коли вони відтворюються в ланцюжку Ethereum Classic. Як показано на рисунку 2.1 [7] , хоча параметри залишаються незмінними, за допомогою атаки повторного відтворення можна здійснити кілька переказів.

```
function transferProxy(address _from, address _to, uint256 _value, uint256 _fee,
    uint8  v, bytes32  r, bytes32  s) public returns (bool){
        bytes32 h = keccak256(_from,_to,_value,_fee);
        if(_from != ecrecover(h,_v,_r,_s)) revert();
    }
```

Рисунок 2.1 – Атака відтворення.

Атака повтору - це мережева атака, при якій дійсні дані транзакції зловмисно або шахрайським шляхом повторюються або затримуються. Це може проявлятися в різних контекстах, включаючи сферу технології блокчейн і, зокрема, Ethereum. За такого сценарію учасник може "програвати" одну й ту саму транзакцію кілька разів, по суті, повторно купуючи товари чи послуги з використанням однієї й тієї ж платіжної інформації.

Інфраструктура блокчейн-транзакцій Ethereum за своєю суттю є вразливою до атак повторного відтворення. Користувач, який підписує повідомлення і завантажує його в контракт, робить інформацію про підпис загальнодоступною, враховуючи прозорість, характерну для блокчейну. Це дає можливість зловмисникам заволодіти підписом. Після перевірки підпису користувача в контракті, якщо підписане повідомлення не містить змінних, які випадковим чином змінюються з кількістю транзакцій, таких як часові мітки або значення nonce, зловмисник може використати цю статичну природу підпису для підробки транзакцій і, відповідно, отримання незаконного прибутку.

Вразливість Ethereum до атак повтору стала особливо помітною в контексті хардфорку, який призвів до появи ланцюжків Ethereum і Ethereum Classic. В ході цієї події було виявлено, що транзакції, призначені і виконані в ланцюжку Ethereum, залишалися дійсними при відтворенні в ланцюжку Ethereum Classic, і навпаки. По суті, це створювало ризик того, що та сама транзакція, яка була підписана і відправлена в одному ланцюжку, може бути перехоплена і повторно відправлена в альтернативному ланцюжку. Система розпізнає таку транзакцію як легітимну, оскільки приватний ключ і nonce користувача дійсні в обох ланцюжках. Щоб усунути цю вразливість, Ethereum запровадив унікальний ідентифікатор для кожної мережі, відомий як "ідентифікатор ланцюжка". Ідентифікатор ланцюга, представлений у Пропозиції щодо вдосконалення Ethereum (EIP) 155, був включений в процес підписання транзакцій. Таким чином, транзакція після EIP-155 буде включати цей ідентифікатор ланцюжка. Введення унікального ідентифікатора ланцюжка для кожної мережі означало, що навіть якщо зловмисник спробує відтворити транзакцію з однієї мережі в іншу, система визнає її недійсною.

Причиною цього є те, що підпис транзакції тепер включатиме ідентифікатор ланцюга, який не відповідатиме ідентифікатору ланцюга альтернативної мережі, що зробить транзакцію недійсною в неправильній мережі.

2.1.2 Атака хибного поповнення рахунку

Поле статусу в квитанції про транзакцію токена Ethereum має значення true або false в залежності від того, чи було згенеровано виключення під час виконання транзакції. Коли користувач викликає функцію переказу контракту токенів для переказу, якщо функція переказу працює нормально і не виникає винятків, статус транзакції дорівнює true. Якщо біржі цифрових валют, гаманці та інші платформи мають недоліки у визначенні успішності транзакцій поповнення токенів, це призведе до серйозних атак хибного поповнення. Як показано на рисунку 2.2 коли balances[msg.sender]< _value, він переходить в логічну секцію else і повертає false, і в результаті не генерується виключення. В цій атаці, хоча біржа не отримала реальних токенів, виконання транзакції не викликало виключення, і користувач отримав реальний запис про поповнення. В цьому випадку користувачі можуть вкрасти реальні активи. Атака помилкового поповнення стала типом атаки, яку неможливо ігнорувати в системі блокчейн.

```
function transfer1 (address _to, uint256 _value) returns (bool) {
    if(_value <= balance[msg.sender] && _value > 0)
    {
        balance[msg.sender] -= _value;
        balance[_to] += _value;
        return true;
    } else
        return false;
}
```

Рисунок 2.2 – Атака хибного поповнення рахунку.

Блокчейн Ethereum використовує унікальну систему отримання транзакцій для виконання взаємодії смарт-контрактів. Ця система включає поле "статус", яке може

бути як істинним, так і хибним, залежно від стану виконання транзакції. Це поле статусу відіграє важливу роль у перевірці успішності транзакції і, відповідно, впливає на безпеку цифрових бірж.

Коли користувач ініціює функцію переказу в контракті токена для передачі активів, виконання транзакції перевіряється за допомогою поля статусу в квитанції про транзакцію токена Ethereum. Якщо функція переказу працює безперебійно, без винятків, статус транзакції позначається як істинний. І навпаки, якщо під час виконання виникає виняток, статус транзакції буде хибним.

Однак цей механізм перевірки відкриває вікно потенційної вразливості, якщо його неправильно реалізувати. Біржі цифрових валют, гаманці та інші платформи повинні точно визначати успішність транзакцій поповнення токенів, інакше це може призвести до атак "хибного поповнення"[13].

Атака "хибного поповнення" - це сценарій, коли, незважаючи на те, що біржа не отримує реальних токенів, виконання транзакції не генерує виключення, в результаті чого відображається справжній статус транзакції. Отже, користувач все одно отримує реальний запис про поповнення. По суті, користувач отримує зараховані активи без реального переказу, що призводить до ситуації, коли користувачі можуть незаконно придбати реальні активи.

Така вразливість виникає через недосконалу реалізацію перевірки успішності транзакцій цифровими біржами та гаманцями. Ці платформи, покладаючись на поле статусу для визначення успішності транзакції, можуть не враховувати нюанси, пов'язані з виконанням транзакції, такі як повторний вхід або атаки на випередження, які не створюють винятків, але все одно порушують запланований потік транзакцій.

2.1.3 Атака на залежність від порядку транзакцій

Атака на залежність від порядку транзакцій - це тип атаки, який широко розповсюджений в системі блокчейн; приклад атаки на залежність від порядку транзакцій показано на рисунку 2.3 У блокчейні транзакції, ініційовані вузлами,

повинні бути упаковані майнерами, перш ніж вони можуть бути остаточно записані в блокчейн. Майнери вибирають серію транзакцій з торгового пулу, а потім упаковують їх у новий блок. Відповідно до критеріїв відбору транзакцій, майнери, як правило, обирають комісію за транзакції для сортування та пакування, щоб отримати максимальну вигоду. Тому послідовність серії транзакцій, упакованих в блок, не збігається з послідовністю генерації транзакцій, але також пов'язана з вартістю газу, спожитого транзакцією. Тому код контракту не може знати порядок транзакцій. Крім того, транзакцію видно кожному вузлу в пулі транзакцій, тому порядок її виконання можна спостерігати.

```
event Purchase(address _buyer, uint256 _price);
event PriceChange(address _owner, uint256 _price);
modifier ownerOnly() {
    require(msg.sender == owner);
    _;
}
function TransactionOrdering() {
    owner = msg.sender;
    price = 100;
}
function buy() returns (uint256) {
    Purchase(msg.sender, price);
    return price;
}
function setPrice(uint256 _price) ownerOnly() {
    price = _price;
    PriceChange(owner, price);
}
```

Рисунок 2.3 – Атака на залежність від порядку транзакцій.

Зловмисник спостерігає за транзакціями, які можуть містити цільовий контракт у пулі. Якщо такі контракти існують, зловмисник змінює статус контракту, який не є сприятливим для зловмисника, або повноваження контракту. Зловмисники також можуть викрадати дані транзакцій, створювати власні транзакції за вищою ціною газу, а потім упаковувати свої транзакції в блок перед оригінальною транзакцією, отримуючи таким чином пріоритет в обробці транзакцій. В Ethereum geth-клієнті txpool складається з двох частин, а саме: черги очікування і черги, що

стоїть в черзі. Якщо Nonce транзакції-відправника більший за Nonce транзакції-відправника + 1, транзакція буде поставлена в чергу, а якщо поточний Nonce транзакції-відправника дорівнює Nonce транзакції-відправника + 1, транзакція буде поміщена в чергу, що очікує на упаковку.

Як невід'ємна частина технології блокчейн, смарт-контракт не тільки розширює сферу застосування технології блокчейн, але й збільшує поверхню атак, з якими стикається система блокчейн. Смарт-контракт пишеться мовою високого рівня, такою як solidity, а потім компілюється в байт-код, який буде розгорнутий в блокчейні власником контракту і запущений на різних віртуальних машинах, подібних до віртуальних машин Ethereum. В процесі роботи смарт-контракт буде стикатися з різними загрозами безпеці .

Блокчейн-системам притаманні властивості, які, будучи невід'ємною частиною їхньої роботи, можуть відкривати шляхи для певних форм експлуатації. Однією з таких властивостей є спосіб обробки транзакцій, який робить їх вразливими до атак на залежність від порядку транзакцій. Ці атаки маніпулюють порядком обробки транзакцій для отримання несанкціонованої вигоди.

У типовій системі блокчейн транзакції, ініційовані вузлами, спочатку повинні бути упаковані майнерами, перш ніж вони можуть бути остаточно записані в блокчейн. Цей процес пакування передбачає, що майнери вибирають серію транзакцій з пулу транзакцій і упаковують їх у новий блок. Ці майнери, мотивовані власними інтересами, зазвичай обирають транзакції з вищою комісією, таким чином максимізуючи свій прибуток.

В результаті, послідовність транзакцій в блоці визначається не порядком, в якому вони були згенеровані, а скоріше такими факторами, як вартість газу, спожитого кожною транзакцією. Такий підхід вносить невизначеність у порядок транзакцій, оскільки код смарт-контракту не може визначити точний порядок транзакцій. Крім того, видимість транзакцій в пулі транзакцій дозволяє кожному вузлу спостерігати за порядком їх виконання [8].

Зловмисник може використати вищезгадані властивості для отримання неправомірної переваги. Зловмисник може спостерігати за транзакціями в пулі, які

можуть взаємодіяти з цільовим контрактом, і діяти відповідно. Зловмисник може змінити статус контракту до невигідного для зловмисника стану або навіть змінити дозволи контракту.

Крім того, зловмисники можуть отримати пріоритет в обробці транзакцій, викрадаючи дані про транзакції, створюючи власні транзакції з вищою ціною на газ і упаковуючи свої транзакції в блок перед оригінальною транзакцією.

Ця вразливість стає ще більш помітною в geth-клієнті Ethereum, де пул транзакцій, txpool, розділений на чергу "pending" і чергу "queued". Транзакція з nonce більшим за завершену транзакцію nonce плюс один ставиться в чергу, в той час як транзакція з nonce, що дорівнює завершеній транзакції nonce плюс один, ставиться в очікуванні, очікуючи на пакування.

2.1.4 Атака на цілочисельне переповнення

Цілочисельне переповнення - типова лазівка в системі блокчейн, яка свого часу спричинила серйозні економічні втрати в розвитку блокчейну. У платформі Ethereum мова Solidity є найбільш поширеною мовою для написання інтелектуальних контрактів. Через незахищеність її конструкції, переповнення цілих чисел є серйозною проблемою. Взагалі кажучи, цілочисельне переповнення можна розділити на цілочисельне переповнення і цілочисельне недоповнення. Відповідно до арифметичної класифікації, існує три проблеми переповнення: переповнення при множенні, переповнення при додаванні та переповнення при відніманні. У квітні 2018 року майже 6 мільярдів юанів було викрадено хакерами через цілочисельне переповнення в контрактному коді американського проекту Chain BEC, що знизило ринкову вартість токенів майже до нуля. У тому ж місяці хакери використали уразливість цілочисельного переповнення на стороні проекту SMT, щоб створити величезну кількість валюти SMT для продажу, а біржа Firecoin призупинила поповнення і виведення всіх інших валют для цієї мети.

У Solidity змінна підтримує беззнакові цілі числа, а значення після uint представляє кількість біт, що займають її беззнакові цілі числа в пам'яті, і підтримує

від 8-бітних беззнакових цілих до 256-бітних беззнакових цілих чисел. Беззнакове число типу uint8 зберігається у діапазоні від 0 до 2^8-1

тобто [0, 255], і ціле беззнакове число типу uint256, яке зберігається у діапазоні від 0 до 2^256-1 . Оскільки діапазон цілих чисел від uint8 до uint256, що зберігаються, обмежений, і діапазон цілих чисел, що виводяться, також обмежений, виникає проблема переповнення. Атака на цілочисельне переповнення показана на рисунку 5 При введенні то виникає недоповнення

```
function withdraw(uint _amount) {
    balancesAndAmount(balances[msg.sender], _amount);
    require(balances[msg.sender] -_amount > 0);
    msg.sender.transfer(_amount);
    balances[msg.sender] -= _amount;
}
```

Рисунок 2.4 – Атака на цілочисельне переповнення.

Solidity, контрактно-орієнтована мова програмування Ethereum, використовує статичну типізацію і підтримує різноманітні типи даних, включаючи цілі числа. Однак, вона накладає певні обмеження на ці типи; наприклад, uint256, беззнакове 256-бітне ціле число, має максимальне значення 2^256-1. Якщо математична операція намагається збільшити це максимальне значення, результат не вкладається у межі вказаного типу даних, що призводить до "переповнення". Значення, по суті, перевертається і починається з нуля, що призводить до некоректного обчислення, яке може мати далекосяжні наслідки.

Аналогічно, "недоповнення" може статися, якщо операція намагається зменшити беззнакове ціле число нижче нуля. Результат обертається до максимально допустимого значення для цього типу, створюючи велике додатне ціле число замість від'ємного, що не підтримується типом беззнакового цілого.

Явища переповнення та недоповнення дають можливість зловмисникам використовувати ці вразливі місця в блокчейні Ethereum. Атака на цілочисельне переповнення передбачає маніпулювання логікою контракту, щоб викликати переповнення або недоповнення, що часто призводить до фінансової вигоди для зловмисника за рахунок інших сторін.

Класичним прикладом є криптовалютний токен-контракт, де зловмиснику вдається ініціювати переказ токенів, який переповнює баланс, що призводить до значного збільшення балансу зловмисника замість його зменшення. Цей тип атаки може призвести до величезних фінансових втрат і підірвати довіру до безпеки та надійності блокчейну Ethereum.

Запобігання атакам цілочисельного переповнення вимагає суворих практик безпеки при розробці та виконанні смарт-контрактів. Безпечні математичні бібліотеки, такі як SafeMath від OpenZeppelin, надають функції, які включають вбудовані перевірки переповнення і недоповнення, генеруючи виключення при виникненні цих подій.

Крім того, автоматизовані інструменти аналізу можуть допомогти виявити потенційні вразливості в коді контракту до його розгортання. Ручний перегляд коду та аудит досвідченими розробниками та експертами з безпеки залишаються найкращою практикою у виявленні потенційних загроз та забезпеченні стійкості до таких атак.

Колективні зусилля спільноти Ethereum, такі як Пропозиції щодо вдосконалення Ethereum (EIP), також пропонують критичні виправлення та оновлення для підвищення безпеки блокчейну. Наприклад, EIP-170, серед інших заходів, встановлює обмеження на розмір контракту, що опосередковано стримує зловмисників від розгортання надмірно складних контрактів, які можуть містити приховані атаки.

2.1.5 Атака на повторний вхід

Атака на повторний вхід є типовою атакою в Ethereum, яка безпосередньо призвела до жорсткої біфуркації Ethereum. Основною причиною атаки є послідовність і атомарність оновлення змінних смарт-контракту і передачі операцій, атака повторного входу показана на рисунку 2.5 [9]. Коли логіка в коді смарт-контракту приймає послідовність спочатку операції передачі, а потім модифікації значення змінної, зловмисник може сконструювати смарт-контракт зі шкідливою

функцією зворотного виклику. Якщо об'єктом операції переказу є зловмисний контракт, це може призвести до рекурсивного виклику контракту, руйнування початкової бізнес-логіки контракту та обходу його перевірки для отримання додаткового доходу від переказу.

```
contract Victim{
    function withDraw(){
        uint amount = userBalannce[msg.sender];
        if (amount > 0) {
            msg.sender.call.value(amount)();
            userBalannce[msg.sender] =0;
        }}
}
contract Attacker{
    function() payable{
        test++;
        Victim(msg.sender).call(bytes4(keccak256("withDraw()")));
    }
}
```

Рисунок 2.5 – Атака на повторний вхід.

За замовчуванням смарт-контракт Ethereum має безіменну функцію зворотного виклику, яка не має параметрів або значень, що повертаються. Якщо у викликаючому контракті не буде знайдено функції, яка б відповідала хешу наданої функції, буде викликана функція зворотного виклику. Коли контракт отримає переказ без даних, він також викличе функцію зворотного виклику. Крім того, щоб отримати Ефір, функція зворотного виклику повинна бути позначена як така, що підлягає оплаті. Якщо вона не позначена як payable, контракт може отримати Ефір, лише викликавши інші функції, позначені як payable. Уявіть собі такий сценарій, якщо побудувати спеціальну функцію зворотного виклику, в якій викликається передавальна функція іншої сторони, то буде згенеровано рекурсивний переказ, і контракт з лазівками буде безперервно переказувати гроші на спеціальний контракт, поки не вичерпається газ. Слід зазначити, що ця атака спрямована тільки на метод переказу address.call.value () в солідарності Ethereum.

Значну загрозу для блокчейну Ethereum становлять атаки на повторний вхід [10]. Ці атаки пов'язані з послідовністю і атомарністю змінних смарт-контрактів і операцій передачі. Хардфорк Ethereum був спровокований значною атакою на

повторний вхід, що ілюструє серйозність цього типу атак. Виявлення механізмів атак на повторний вхід та розробка ефективних заходів протидії має вирішальне значення для безпеки Ethereum.

Атака на повторний вхід передбачає використання контракту, який слідує логічній послідовності "спочатку переказ, потім оновлення змінної". Ця атака використовує шкідливу функцію зворотного виклику, вбудовану в смарт-контракт. Коли операція переказу спрямована на зловмисний контракт, можуть бути ініційовані рекурсивні виклики контракту в обхід системи стримувань і противаг, що призведе до ненавмисного збільшення прибутку від переказу для зловмисника.

За замовчуванням смарт-контракти Ethereum містять безіменну резервну функцію без параметрів та значень, що повертаються. Ця функція викликається, коли в викликаючому контракті не знайдено відповідного хешу функції або коли контракт отримує переказ без будь-яких супровідних даних. Щоб отримати Ефір, резервна функція повинна бути позначена як "payable". Якщо вона не позначена, то контракт може отримувати Ефір тільки через інші функції, позначені як "payable".

Атака повторного входу може статися, коли спеціальна резервна функція викликає функцію переказу іншої сторони, що призводить до рекурсивних переказів. Вразливий контракт продовжує переказувати кошти на спеціальний контракт, поки газ не вичерпається. Примітно, що цей тип атаки є специфічним для методу переказу Ethereum Solidity address.call.value().

Серйозність атак на повторний вхід стала очевидною після інциденту, який призвів до хардфорку Ethereum. Ці атаки порушують заплановану бізнес-логіку контрактів і обходять перевірки, тим самим потенційно призводячи до фінансових втрат і підриваючи загальну довіру до блокчейну.

Для захисту від атак повторного входу розробники Ethereum можуть використовувати такі методи, як патерн "перевірки-ефекти-взаємодії", який передбачає перенесення будь-яких зовнішніх викликів в кінець функції після завершення всієї внутрішньої роботи. Крім того, для блокування повторних викликів можуть використовуватися засоби захисту від повторного входу, такі як м'ютекси.

2.1.6 Атака на ханіпот

Найцікавішими знахідками є контракти "ханіпот". Ці контракти зберігають ефір, але роблять вигляд, що зберігають його ненадійно. Коротше кажучи, це шахрайські контракти, які намагаються обдурити нас, щоб ми думали, що можемо вкрасти ефір, який вони зберігають, тоді як насправді все, що ми можемо зробити - це втратити ефір. Як показано на Рисунку 2.6 [11], крипто-рулетка є різновидом атаки "медового горщика". Змінна game не ініціалізована, тому за замовчуванням вона вказує на перше місце зберігання контракту, а потім зберігає тут адресу абонента. Переданий номер зберігається у другій комірці. Фактично, змінна secretNumber з часом перезаписується адресою абонента. Загальна схема, якої вони дотримуються, полягає в тому, що для того, щоб виграти ефір, яким вони володіють, ми повинні спочатку відправити їм свій власний ефір. Однак, якщо ми спробуємо це зробити, на нас чекає неприємний сюрприз: смарт-контракт з'їдає наш ефір, і ми виявляємо, що смарт-контракт робить не те, що ми думали, що він буде робити.

```
struct Game {
    address player;
    uint256 number;
}
Game[] public gamesPlayed;
function shuffle() {
    secretNumber = uint8(sha3(now, block.blockhash(block.number-1)) )% 10 ;
}
function play(uint256 number) payable public {
    require(msg.value >= betPrice && number <= 10);
    Game game;
    game.player = msg.sender;
    game.number = number;
    gamesPlayed.push(game);
    if (number == secretNumber) {
        msg.sender.transfer(this.balance);
    }
    shuffle();
    lastPlayed = now;
}
```

Рисунок 2.6 – Атака на ханіпот.

Контракти типу "ханіпот" представляють собою захоплююче, але водночас підступне явище в блокчейні Ethereum. Ці контракти навмисно створюють ілюзію ненадійного зберігання Ефіру, спокушаючи користувачів повірити, що вони можуть вкрасти кошти, які в ньому містяться. Однак реальність далека від початкового враження. Потрапляння в пастку контракту "ханіпот" часто призводить до втрати Ефіру, а не до будь-якої фінансової вигоди.

Крипторулетка є яскравим прикладом honeypot-атаки. Ці контракти експлуатують користувачів, заробляючи на неініціалізованих змінних і обманному розподілі сховища. У випадку з CryptoRoulette змінна "game", якщо її не ініціалізувати, вказує на перше місце в пам'яті контракту, де зберігається адреса абонента. У другій комірці зберігається переданий номер. Таким чином, змінна 'secretNumber', яка повинна представляти виграшний номер, ненавмисно перезаписується адресою абонента.

Для подальшого обману користувачів, контракти ханіпотt зазвичай діють за певною схемою. Користувачів спокушають відправити певну кількість власного Ефіру як передумову для виграшу Ефіру, що зберігається в рамках контракту. Однак, здійснивши цю транзакцію, користувачі стикаються з суворим усвідомленням того, що смарт-контракт зловмисно поглинає їхній Ефір, залишаючи їх з порожніми руками і розчарованими.

Honeypot-контракти створюють значні ризики для користувачів, які нічого не підозрюють, зловживаючи їхньою довірою та експлуатуючи їхнє прагнення до швидкої наживи. Використовуючи неініціалізовані змінні та оманливий розподіл сховища, ці контракти хитро вводять користувачів в оману, змушуючи їх повірити, що вони знайшли можливість для отримання прибутку.

Оманливі стратегії, що застосовуються в контрактах-"ханіпот", вимагають від користувачів підвищеної обізнаності та всебічного розуміння функціональності смарт-контрактів. Користувачі повинні проявляти обережність і проводити ретельну перевірку перед тим, як вступати в контакт з будь-яким контрактом Ethereum, щоб не потрапити ненавмисно в пастку схеми "ханіпот".

2.1.7 Атака за коротким URL-адресом

Атака на короткі URL-адреси - типова атака в Ethereum, яка зазвичай відбувається на біржах. У віртуальній машині Ethereum кінець вхідних даних автоматично заповнюється 0. Зловмисники можуть використовувати адресний акаунт з кінцем 0, а біржа не перевіряє довжину адреси, введеної користувачем, що призводить до зсуву і збільшення пов'язаних змінних, що передаються, таким чином збільшуючи фактичну суму переказу в кілька разів, і зловмисники можуть отримати велику кількість вигоди. Існує дві основні причини цієї вразливості: перша - біржа не перевірила довжину вхідної адреси користувача, а друга - віртуальна машина Ethereum має механізм автоматичного завершення даних, довжина яких не відповідає специфікації, при виклику смарт-контракту, що призводить до посилення зсуву параметрів. Ми можемо використовувати sendRawTransaction() для здійснення цієї атаки, код якої показано на рисунку 2.7.

```
mapping (address => uint) balances;
event Transfer(address indexed _from, address indexed _to, uint256 _value);
function transfer(address to, uint amount) public returns(bool success) {
    if (balances[msg.sender] < amount) return false;
    balances[msg.sender] -= amount;
    balances[to] += amount;
    emit Transfer(msg.sender, to, amount);
    return true;
}
```

Рисунок 2.7 – Атака за коротким URL-адресом

Атака на короткі URL-адреси - це добре відомий тип вразливості, яка спеціально націлена на біржі Ethereum. Використовуючи механізм автоматичного заповнення даних EVM та відсутність перевірки довжини адреси на біржах, зловмисники можуть маніпулювати процесом переказу на свою користь. Це призводить до значного збільшення фактичної суми переказу, що дозволяє зловмисникам отримати значну вигоду. Вразливість полягає в тому, що EVM автоматично доповнює дані нулями, коли довжина не відповідає специфікації. Зловмисники користуються цим, використовуючи адресний обліковий запис з кінцевим нулем. Біржі, не перевіряючи довжину вхідної адреси користувача,

ненавмисно спричиняють зсув і розширення змінних, пов'язаних з переказом. Як наслідок, сума переказу збільшується в кілька разів, що дає зловмиснику значну перевагу.Атака за короткою URL-адресою використовує дві основні вразливості. По-перше, біржі зазвичай не мають належних механізмів перевірки довжини вхідних адрес користувачів. Цей недогляд дозволяє зловмисникам маніпулювати змінними і отримувати неочікувані суми переказів. По-друге, механізм автоматичного заповнення даних EVM, який заповнює кінець даних нулями, сприяє збільшенню параметрів і ще більше посилює вплив атаки.Атака за короткою URL-адресою може бути виконана за допомогою функції sendRawTransaction(). Створюючи транзакцію з ретельно підібраними змінними і параметрами, зловмисники можуть використовувати вразливості в системі перевірки довжини адреси біржі і механізмі заповнення даних EVM. Це дозволяє їм ініціювати перекази зі значно завищеними сумами, що приносить зловмиснику значну вигоду.

2.1.8 Атака запису довільної адреси пам'яті

Атака запису довільної адреси пам'яті є поширеною і шкідливою атакою в системі блокчейн. Атака може призвести до того, що зловмисники запишуть і перезапишуть будь-яку змінну зберігання в смарт-контракті. В Ethereum змінні стану інтелектуальних контрактів будуть зберігатися в області зберігання, яка є важливим і відкритим місцем зберігання контрактів. Взагалі кажучи, розробники контрактів встановлюють суворий контроль доступу до глобальних змінних, що зберігаються в області зберігання, щоб забезпечити безпеку контрактів. Для зберігання даних використовується відображення пари ключ-значення. Якщо користувач може довільно контролювати значення ключа сховища під час запису, він може змінити будь-яке значення змінної сховища, щоб уникнути всіх пов'язаних з цим операцій виявлення в контракті, які використовують значення змінної стану для перевірки повноважень, і таким чином досягти мети поліпшення повноважень [12]. Крім того, оскільки зловмисник може використати цю вразливість для руйнування структури зберігання контракту і виконати будь-яку операцію

перезапису змінної, наприклад, перезаписати значення змінної стану, що зберігає адресу власника контракту, це може призвести до ненормального виконання функцій контракту, заморожування коштів та інших небезпек. Оскільки необхідний захист недійсний, власник контракту може спробувати вийти за межі масиву, виконавши код на рисунку 2.8 [13], коли довжина масиву bonusCodes дорівнює 0. Таким чином, ми можемо писати в будь-яке місце в сховищі довільним чином.

```
function PopBonusCode() public {
    require(0 <= bonusCodes.length);
    bonusCodes.length--;
}
function UpdateBonusCodeAt(uint idx, uint c) public {
    require(idx < bonusCodeslength);
    bonusCodes[idx] = c;
}
```

Рисунок 2.8 – Атака запису довільної адреси пам'яті

Атака довільного запису адреси пам'яті є поширеною і шкідливою вразливістю безпеки в системах блокчейн. Ця атака дозволяє зловмисникам маніпулювати змінними зберігання в смарт-контрактах, потенційно ставлячи під загрозу цілісність і безпеку всієї системи. Наприклад, Ethereum покладається на область сховища для зберігання змінних стану, що робить її критично важливим і відкритим місцем для зберігання даних. Щоб забезпечити безпеку контракту, розробники зазвичай впроваджують суворий контроль доступу до глобальних змінних, що зберігаються в цій області. Змінні стану смарт-контракту зберігаються в області зберігання, використовуючи структуру відображення пари ключ-значення. Зазвичай розробники контрактів застосовують суворі заходи контролю доступу до глобальних змінних, що зберігаються в цій області, для підтримки безпеки контракту. Однак, якщо користувач отримує можливість довільно керувати парами ключ-значення під час зберігання, він може змінити значення будь-якої змінної, ефективно обходячи будь-які операції виявлення та перевірки авторизації. Ця маніпуляція дозволяє зловмиснику підвищити свій авторитет і потенційно поставити під загрозу безпеку контракту. Атака довільного запису адреси пам'яті створює кілька значних ризиків в системах блокчейн. Зловмисники можуть використовувати цю вразливість, щоб порушити структуру зберігання контракту і

перезаписати критичні змінні, такі як адреса власника контракту. Це може призвести до неправильного виконання функцій контракту, заморожування коштів та інших небезпечних наслідків. Зробивши необхідні засоби захисту неефективними, власник контракту може навіть спробувати переповнити розмір масиву, що ще більше посилить наслідки атаки.

Щоб зменшити ризики, пов'язані з атакою запису довільної адреси пам'яті, необхідно впровадити механізми суворого контролю доступу. Розробники, що працюють за контрактом, повинні впроваджувати надійні перевірки авторизації та ретельно керувати зіставленнями "ключ-значення", щоб запобігти несанкціонованим модифікаціям. Крім того, використання безпечних методів кодування та проведення ретельного аудиту коду може допомогти виявити та усунути вразливості, які можуть призвести до таких атак.

2.2 Аналіз інцидентів

Аналіз інцидентів дозволяє розробникам виявити та зрозуміти вразливості в системі безпеки, які призвели до порушення. Таке розуміння має вирішальне значення для усунення цих вразливостей і захисту контракту від подібних атак. Аналіз інцидентів передбачає ретельне вивчення інцидентів безпеки, щоб зрозуміти їх походження, вплив і вразливості, які вони використовували. У контексті смарт-контрактів аналіз інцидентів може виявити причини порушень контрактів, вразливості, які були використані, і потенційні заходи для запобігання подібним інцидентам у майбутньому. В ході даної роботи були розглянуті наступні інциденти:

2.2.1 Атака наThe DAO

The DAO - децентралізована автономна організація, у 2015 році спільнота Ethereum почала обговорювати DAO. Концепція цих спільнот на основі блокчейну полягала в тому, що вони могли б координувати людські зусилля через виконання коду, який можна перевірити (смарт-контракти, що працюють на блокчейні Ethereum), та

децентралізоване прийняття рішень щодо протоколів спільноти. У 2016 році було створено DAO під назвою "The DAO". Це був децентралізований, контрольований спільнотою інвестиційний фонд, який залучив 150 мільйонів доларів США (приблизно 3,54 мільйона ETH) шляхом продажу власного токена спільноти[14].

Однак менш ніж через три місяці після запуску The DAO був атакований зловмисником. Протягом наступних тижнів зловмисник викачав зі смарт-контракту The DAO більшу частину ETH на суму 150 мільйонів доларів США за допомогою атаки на повторний вхід. Ця атака призвела до значного порушення роботи The DAO, підриву довіри інвесторів та значного удару по довірі до Ethereum.

Атака на повторний вхід використовує спосіб роботи "резервних" функцій. Запасні функції - це спеціальні конструкції в Solidity, які спрацьовують в певних ситуаціях. Серед особливостей резервних функцій - безіменність, виклик ззовні, обмежена кількість до нуля або однієї на контракт, автоматичний запуск, коли інший контракт викликає функцію в охоплюючому смарт-контракті резервної функції, і може включати довільну логіку. Атака повторного входу використовує п'яту і шосту особливості резервних функцій і покладається на певний порядок операцій в контракті-жертві.

Іншими словами, хтось може рекурсивно вийти з DAO, виводячи суми, що дорівнюють їх початковим інвестиціям в ETH, на невизначений термін, ще до того, як запис про їх виведення буде зафіксовано в оригінальному контракті DAO.

Ось вразливість, виявлена в файлі контракту Solidity DAO.sol:

```
function splitDAO(
uint _proposalID,
address _newCurator
) noEther onlyTokenholders returns (bool _success) {
...
// [Added for explanation] The first step moves Ether and assigns new tokens
uint fundsToBeMoved =
(balances[msg.sender] * p.splitData[0].splitBalance) /
p.splitData[0].totalSupply;
```

if (p.splitData[0].newDAO.createTokenProxy.value(fundsToBeMoved)(msg.sender) == false) //

[Added for explanation] This is the line that splits the DAO before updating the funds in the account calling for the split

...

// Burn DAO Tokens

Transfer(msg.sender, 0, balances[msg.sender]);

withdrawRewardFor(msg.sender); // be nice, and get his rewards

// [Added for explanation] The previous line is key in that it is called before

totalSupply and balances[msg.sender] are updated to reflect the new balances after the split

has been performed

totalSupply -= balances[msg.sender]; // [Added for explanation] This happens after the split

balances[msg.sender] = 0; // [Added for explanation] This also happens after the split

paidOut[msg.sender] = 0;

return true;

}

Як показано тут, DAO посилається на масив балансів, щоб визначити, скільки токенів DAO доступні для переміщення. Значення p.splitData[0] є властивістю пропозиції, що подається до DAO, а не будь-якою властивістю DAO. Це, в поєднанні з тим, що withdrawRewardFor викликається до оновлення balances[], дало можливість зловмиснику викликати fundsToBeMoved необмежену кількість разів, тому що їх баланс все одно повернеться до початкового значення.

Якщо уважніше придивитися до withdrawRewardFor(), то можна побачити умови, які зробили це можливим:

function withdrawRewardFor(address _account) noEther internal returns (bool _success) {

if ((balanceOf(_account) * rewardAccount.accumulatedInput()) / totalSupply < paidOut[_

```
account])
throw;
uint reward =
(balanceOf(_account) * rewardAccount.accumulatedInput()) / totalSupply - paidOut[_
account];
if (!rewardAccount.payOut(_account, reward)) // [Added for explanation] this is the
statement that is vulnerable to the recursion attack. We must go deeper.
throw;
paidOut[_account] += reward;
return true;
}
```

Якщо припустити, що перший оператор буде оцінено як false, то буде виконано оператор, позначений як вразливий. Залишився ще один крок, щоб зрозуміти, як зловмисник зміг це зробити. Перший раз, коли функція withdrawRewardFor (коли зловмисник мав законні кошти для виведення), перший оператор буде коректно оцінюється як false, що призводить до запуску наступного коду:

```
function payOut(address _recipient, uint _amount) returns (bool) {
if (msg.sender != owner || msg.value > 0 || (payOwnerOnly && _recipient != owner))
throw;
if (_recipient.call.value(_amount)()) { // [Added for explanation] this is the coup de
grace
PayOut(_recipient, _amount);
return true;
} else {
return false;
}
splitDao
withdrawRewardFor
payOut
recipient.call.value()()
```

splitDao

withdrawRewardFor

payOut

recipient.call.value()()

Таким чином, зловмисник зміг виводити кошти з DAO в дочірню DAO на невизначений час.

Нагадаємо, що зловмисник зробив наступне:

1. Розділив DAO.

2. Вивести кошти в нову DAO.

3. Рекурсивно викликати функцію розділення DAO до того, як код перевірить чи доступні кошти.

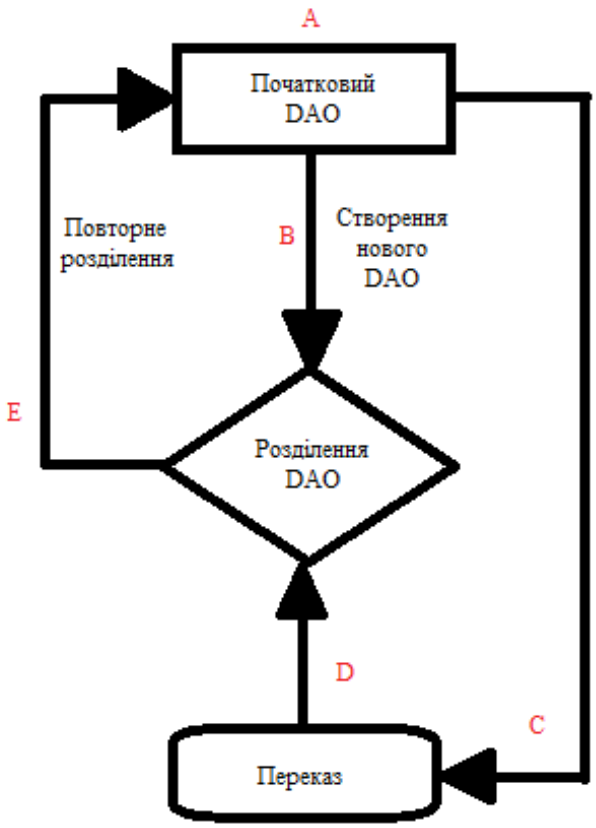

Рисунок 2.9 – Процес атаки на The DAO

В результаті зловмисник зміг викрасти близько 3,6 мільйона ETH, що на момент атаки становило близько 50 мільйонів доларів США.

Інвестори DAO опинилися в особливо вразливому становищі. Мало того, що The DAO була скомпрометована, але якби вони спробували вивести кошти у власну дочірню DAO, то отриманий контракт мав би ті ж самі вразливості, що й оригінальний.

Інвестори DAO були не єдиними, хто був зацікавлений у результаті такого повороту подій. Ажіотаж навколо DAO досяг релігійних масштабів, які Бутерін передбачав ще у 2014 році. Майже 5 відсотків ETH, що перебували в обігу на той час, було інвестовано в DAO. Це мало ряд наслідків для всієї екосистеми Ефіріуму і призвело до однієї з найбільш запеклих дискусій в короткій історії блокчейну.

З одного боку дебатів були ті, хто прагнув захистити молоду екосистему Ethereum від зловмисників, що володіють зловмисника, який володіє нетривіальною частиною загального обсягу ETH в обігу. Вони не обов'язково були стурбовані тим, чи чи виживе The DAO, але в кінцевому підсумку хотіли переконатися, що Ethereum виживе як авторитетна блокчейн-платформа, на якій в майбутньому можуть бути побудовані інші ДАО. Це було Бутерін і багато хто з основних членів команди розробників Ethereum були налаштовані саме так.

З іншого боку були ті, хто був відданий ідеалам децентралізації та незмінності. В очах багатьох в цьому таборі (назвемо їх табором справедливості), блокчейн є за своєю суттю справедливою системою, оскільки вона є детермінованою, і кожен, хто вирішує використовувати її, неявно погоджується з цим фактом. У цьому сенсі зловмисник DAO зловмисник не порушив жодних законів. Навпаки, атака реентерації використовувала програмний код, який складав внутрішній регламент DAO, і налаштувала його проти нього самого.

Табір децентралізації вважав, що перезапис блокчейну для відкату зловмисника секвестр ETH в дочірніх DAO поставило б під загрозу цілісність блокчейну. Блокчейн, згідно з цією точкою зору, повинен був бути незмінним і без будь-якого центрального органу влади, включаючи Фонд Ефіріума. Вони були стурбовані моральним ризиком, пов'язаним з тим, що невелика група людей переписування блокчейну може відкрити двері для інших втручань, таких як вибіркова цензура.

Обидві сторони палко обговорювали свої позиції в соціальних мережах і в засобах масової інформації. Цей процес прославив концепції м'яких і жорстких форків.

Розгалуження блокчейнів - або будь-якого програмного коду з цього приводу - не було чимось новим для Ethereum. чи будь-якого програмного коду - не є чимось новим для Ethereum або DAO, але це стало центром дебатів між табором справедливості та табором незмінності.

Тим часом група хакерів "білих капелюхів" працювала цілодобово, намагаючись зламати хакер. Група білих капелюхів складалася з людей як за, так і проти хардфорку, але вони працювали разом, тим не менш, вони працювали разом, щоб виконати деякі з тих же атак, які були виявлені до 17 червня, щоб перевести вкрадені ETH в нові контракти в надії повернути його законним власникам.

Команда білих капелюхів звернулася до людей, які зробили значні інвестиції в The DAO, щоб зібрати грошей для атак переслідування, в ході яких вони могли переслідувати зловмисника в нових DAO з більшими коштами, ніж він зміг вивести зловмисник зміг вивести, що давало їм право більшості голосів у новоствореній DAO.

30 липня понад 90 відсотків хешрейту підтримали форк. Кошти DAO були повернуті інвесторам інвесторам так, ніби організації ніколи не існувало. Щось на кшталт того.

Опозиція хардфорку призвела до появи Ethereum Classic (ETC), оскільки невелика частина спільноти продовжувала майнити оригінальний блокчейн Ethereum. Ці фундаменталісти незмінності були були віддані ідеї, що блокчейн представляє собою нову, революційну модель управління. Не дивлячись на завзятість цієї активної меншості в спільноті Ефіріуму, багато хто не очікував, що обидві версії блокчейну що обидві версії блокчейну виживуть у довгостроковій перспективі. Найбільші біржі та криптосервіси додали підтримку ETC, але багато хто скептично ставився до довгострокових перспектив платформи, яка по суті дублювала можливості Ethereum.

2.2.2 Атака на The Parity

Parity - це цифровий гаманець для блокчейну Ethereum та інших цифрових активів.Контракт Parity Multisig WalletLibrary, розгорнутий через день після першого злому Parity Wallet Hack, містив помилку, яка дозволяла будь-кому виконати функцію

initWallet. Ця вразливість була використана, коли особа, експериментуючи з попереднім експлоітом, викликала initWallet, а потім функцію kill на WalletLibrary, фактично видаливши її з блокчейну. В результаті фактичний контракт гаманця, який делегує всі виклики жорстко закодованому контракту WalletLibrary, втратив логіку, необхідну для переказу коштів. Як наслідок, приблизно 513 000 ETH (еквівалент близько 154 мільйонів доларів США) виявилися заблокованими в контрактах, що постраждали. Примітно, що кошти не були вкрадені, вони просто стали недоступними.

Злом Parity Wallet Hack 1 полягав у тому, що хакер змінив стан різних контрактів гаманця, делегувавши виклик initWallet, встановивши себе як власника контракту гаманця, а потім виводив кошти в звичайному режимі. На відміну від цього, другий злом не змінив внутрішній стан розгорнутих контрактів Wallet; натомість він змінив внутрішній стан контракту WalletLibrary. Контракт WalletLibrary містить змінну стану m_numOwners (разом з m_owners і подібними), яка, як очікується, буде затінена власним станом викликаючого контракту. Решта його стану глобально поділяється між усіма Parity Multisig Гаманцями, які жорстко кодують його адресу.

Контракт WalletLibrary був розгорнутий після Parity Hack 1. Після розгортання контракт WalletLibrary був просто неініціалізований. m_numOwners було 0. Це означало, що модифікатор only_uninitialized, який буде коректно працювати при виклику в контексті ініціалізованого контракту Wallet, завжди передавався. Якщо WalletLibrary не виконувалася в контексті контракту Wallet, m_numOwners дорівнював 0, що дозволяло будь-кому викликати методи, які захищає цей модифікатор, одним з яких є initWallet.

Кошти, заблоковані у відповідних контрактах, наразі недоступні. Можливі підходи для відновлення коштів включають хардфорк, подібний до DAO Hack, злам криптографії і розгортання нового контракту на жорстко закодовану адресу WalletLibrary (що наразі криптографічно неможливо), або впровадження EIP156, який забезпечує контрольований спільнотою спосіб відновлення коштів під час подібних подій.

Основна причина помилки, схоже, полягає в двох факторах: швидко виправлений код WalletLibrary не був перевірений після першого злому Parity Wallet Hack, і шаблон

бібліотеки приховував той факт, що WalletLibrary є фактичним контрактом з внутрішнім станом. Це спонукало до припущення, що вона буде викликатися тільки в контексті контракту з гаманцем. Конструкція бібліотеки Solidity зробила б цю помилку набагато очевиднішою; контрактам бібліотеки Solidity не дозволяється мати внутрішній стан, тому той факт, що будь-хто міг би викликати initWallet, був би набагато очевиднішим. Крім того, вони не змогли б реалізувати власність (внутрішній стан), що робило цей код безпечним; очікувалося, що модифікатор onlyOwners буде працювати коректно.

На закінчення, злом Parity Wallet Hack 2 служить суворим нагадуванням про важливість ретельного аудиту коду і про потенційні пастки певних шаблонів дизайну при розробці смарт-контрактів. Ситуація все ще розвивається, і від Parity очікуються подальші оновлення.

2.2.3 Атака на The Bancor

Bancor - це децентралізована мережа ліквідності, яка дозволяє користувачам тримати будь-який токен і конвертувати його в будь-який інший токен в мережі, без контрагента, за автоматично розрахованою ціною. Протокол Bancor використовує технологію смарт-контрактів для реалізації методу, заснованого на резервах, який може бути інтегрований в будь-який блокчейн, що підтримує цю функціональність. Токен Bancor Network (BNT) служить мережею-хабом, що з'єднує всі токени в мережі Bancor, дозволяючи їм конвертуватися один в одного завдяки їх ліквідності в BNT.

9 липня 2018 року мережа Bancor була скомпрометована, коли один з її акаунтів було зламано. Зловмисники отримали контроль над гаманцем, який використовувався для оновлення деяких смарт-контрактів Bancor, а потім використали його для виведення 24 984 ETH (близько $12,5 млн), 229 356 645 NPXS (близько $1 млн) і 3 200 000 BNT (близько $10 млн). Точний метод, за допомогою якого зловмисники отримали ключ, залишається невідомим, але теорії припускають можливий пролом у внутрішній мережі Bancor або фішингову атаку на один з комп'ютерів розробників мережі[12].

Злом Bancor був складною операцією, яка використовувала вразливість в системі смарт-контрактів мережі Bancor. Зловмисники змогли отримати контроль над гаманцем, який використовувався для оновлення деяких смарт-контрактів Bancor. Цей гаманець мав доступ до ряду критично важливих і обмежених функцій, до яких не мали доступу звичайні акаунти.

Скомпрометований акаунт був початковим творцем токен-контракту Bancor, поширеного шаблону розробки, який використовується багатьма смарт-контрактами в мережі Ethereum. Цей акаунт мав доступ до ряду критично важливих і обмежених функцій, до яких звичайні акаунти не мали доступу. Хоча скомпрометований акаунт більше не був власником жодного критично важливого контракту, його використовували для оновлення деяких супутніх контрактів, пов'язаних з Bancor. З кожного контракту, до якого мав доступ зламаний акаунт, були виведені кошти.

Точний метод, за допомогою якого зловмисники отримали ключ, залишається невідомим, але теорії припускають можливий пролом у внутрішній мережі Bancor або фішингову атаку на один з комп'ютерів розробників мережі. Потім зловмисники використали скомпрометований обліковий запис для ініціювання процедури обміну Bancor і авторизації переказу токенів зі скомпрометованих контрактів на свої власні рахунки.

У відповідь на атаку Bancor вжив низку контрзаходів. Їм вдалося повернути вкрадені токени BNT на суму близько $10 млн, перевести право власності на всі контракти зі зламаного акаунта до більш захищених власників, а також підвищити безпеку свого основного контракту, надавши право власності на нього контракту з декількома підписами. Цей захід гарантує, що жоден акаунт не зможе отримати доступ до всієї мережі. Однак саме ці функції, які допомогли зберегти всі вкрадені токени BNT, можуть бути використані для знищення всієї мережі, якщо два з чотирьох ключів облікових записів потраплять в чужі руки.

Злом Bancor служить нагадуванням про те, що кожна система має свої вразливості, і навіть незначна вразливість може призвести до величезних втрат, якщо нею скористаються завзяті хакери. Однак швидкі та продумані контрзаходи можуть зменшити наслідки будь-якої атаки. Цей інцидент підкреслює важливість суворих

заходів безпеки, включаючи використання контрактів з декількома підписами, регулярний аудит і розробку надійних протоколів відновлення. Він також підкреслює необхідність постійних досліджень і розробок для виявлення і усунення потенційних вразливостей в технології блокчейн і криптовалютних біржах.

Насамкінець, злам Bancor дає цінну інформацію про виклики та ризики, пов'язані з управлінням цифровими активами. Він підкреслює необхідність надійних заходів і протоколів безпеки, а також вказує на потенційні вразливості, притаманні технології блокчейн і криптовалютним біржам. Оскільки використання цих технологій продовжує зростати, дуже важливо, щоб ці уроки були враховані для запобігання подібним інцидентам у майбутньому.

2.2.4 Атака на The bZx Protocol

bZx - це децентралізований протокол, який дозволяє кредитувати і торгувати з маржею і кредитним плечем. Він побудований на Ethereum та інтегрований з децентралізованими біржами Kyber та Uniswap. Протокол покликаний полегшити кредитування, запозичення і маржинальну торгівлю в децентралізованому режимі, усуваючи потребу в посередниках.

Злом bZx був виконаний у п'ять окремих етапів: Flashloan Borrow, Hoard, Margin Pump, Dump і Flashloan Repay. Зловмисник ініціював експлойт, скориставшись функцією флеш-позики dYdX, щоб позичити 10 000 ETH. Отримавши позику, зловмисник поклав 5500 ETH в Compound в якості застави, щоб позичити 112 WBTC. Ці накопичені WBTC були згодом викинуті на четвертому етапі.

На третьому етапі, Margin Pump, зловмисник скористався помилкою в реалізації смарт-контракту bZx. Зловмисник вніс 1300 ETH і викликав функцію маржинальної торгівлі bZx, яка за допомогою KyberSwap обміняла позичені 5 637,623762 ETH на 51,345576 WBTC. Цей обмін підвищив обмінний курс 1 WBTC до 109,8 WETH, що приблизно втричі перевищує звичайний обмінний курс (~38,5 WETH/WBTC).

На четвертому етапі, Dump, зловмисник продав позичені у Compound 112 WBTC назад за WETH в Uniswap, в результаті чого отримав 6 871,4127388702245 ETH із

загальним обмінним курсом 1WBTC=61,4 WETH. Нарешті, на кроці Flashloan Repay зловмисник повернув 10 000.000000000011ETH назад на dYdX, таким чином завершивши флеш-позику.

Експлойт був реалізований завдяки прихованій помилці в реалізації смарт-контракту bZx. Маржинальна накачка почалася з функції marginTradeFromDeposit(). Функція викликає _borrowTokenAndUse() з четвертим параметром, встановленим як true. Усередині _borrowTokenAndUse() викликається _getBorrowAmountAndRate(), коли amountIsADeposit має значення true. Повернута borrowAmount буде збережена в sentAmounts[1]. Також у _borrowTokenAndUse() sentAmounts[6] заповнюється значенням sentAmounts[1] у випадку, якщо amountIsADeposit == true. Пізніше викликається _borrowTokenAndUseFinal().

У функції _borrowTokenAndUseFinal() через інтерфейс IBZx викликається takeOrderFromiToken() так, щоб транзакція перетікала в bZxContract. У функції bZxContract::takeOrderFromiToken() є виклик require(), який перевіряє, чи є позиція здоровою чи нездоровою. Однак, у випадку loadDataBytes.length == 0 && sentAmounts[6] == sentAmounts[1], перевірка на здоровість bZxOracle::shouldLiquidate() буде пропущена. Це саме та умова, за якої спрацьовує експлойт, щоб уникнути перевірки на адекватність.

Якщо розглянути bZxOracle::shouldLiquidate(), то перевірка getCurrentMarginAmount() <= loanOrder.maintenanceMarginAmount зробила б свою роботу, перехопивши крок накачування маржі і таким чином запобігши цій атаці. Однак, через баг в смарт-контракті, ця перевірка була обійдена, що дозволило зловмиснику виконати експлойт.

Злом bZx підкреслює важливість ретельного аудиту смарт-контрактів і потенційні пастки певних шаблонів дизайну при розробці смарт-контрактів. Він також підкреслює витонченість атак в просторі DeFi, де зловмисники експлуатують складну взаємодію між декількома протоколами. Оскільки використання цих технологій продовжує зростати, дуже важливо, щоб ці уроки були враховані для запобігання подібних інцидентів у майбутньому. Цей інцидент слугує нагадуванням про потенційні вразливості протоколів DeFi і необхідність вжиття надійних заходів безпеки,

включаючи використання контрактів з декількома підписами, регулярний аудит і розробку надійних протоколів відновлення. Він також підкреслює необхідність постійних досліджень і розробок для виявлення і усунення потенційних вразливостей в технології блокчейн і протоколах DeFi.

2.2.5 Атака на The Wormhole Bridge

Wormhole - це децентралізований протокол, який дозволяє здійснювати міжмережеві перекази активів між різними блокчейнами. Він побудований на основі Solana та інтегрований з Ethereum, що дозволяє користувачам безперешкодно переміщувати активи між цими двома блокчейнами. Протокол призначений для децентралізованого переказу коштів між ланцюжками, усуваючи потребу в посередниках.

Злом мосту Wormhole Bridge був здійснений зловмисником, який запустив кілька атак з метою обійти процес верифікації мосту Wormhole на Солані. Зловмисник здійснив другу за величиною крадіжку криптовалюти з протоколу DeFi, що призвело до втрати приблизно 120 000 Wormhole Ethereum (WeETH) на суму понад $320 млн.

Зловмисник ініціював експлойт, отримавши 0,94 ETH від Tornado Cash, міксера на базі Ethereum, який був використаний для оплати газових платежів за транзакціями, що відбулися одразу після першого злому. Зловмисник також надіслав 0,1 ETH на депозитну адресу великої міжнародної біржі.

Під час атаки хакер обійшов етап верифікації, ввівши фальшивий обліковий запис sysvar, і успішно згенерував шкідливе "повідомлення", в якому було вказано 120 000 ETH, що підлягають видобутку. Викликавши функцію "complete_wrapped" зі шкідливим "повідомленням", зловмисник успішно видобув 120 000 wETH. Через дві хвилини після майнінгу зловмисник перевів 10 000 ETH в блокчейн Ethereum, а ще через 20 хвилин в блокчейні Ethereum відбулася транзакція на суму 80 000 ETH. До цього дня ці кошти все ще знаходяться в гаманцях зловмисників, в тому числі: CxegPrfn2ge5dNiQberUrQJkHCcimeR4VXkeawcFBBka та 0x629..., які є одними з найбільших серед інших[15].

Зловмисник викликав функцію "verify_signatures" з підробленим обліковим записом sysvar. Підроблена функція "verify_signatures" зі шкідливим "обліковим записом sysvar" використовувалася для обходу процесу верифікації. Функція "load_current_index" не перевіряє, чи введений "обліковий запис sysvar" насправді є "системним sysvar". Оскільки поточна інструкція, отримана з "sysvar", контролюється зловмисником, він зможе успішно виконати наступну перевірку.

Потім зловмисник викликав функцію "post_vaa" з перевіреними підписами з попереднього кроку і створив обліковий запис зловмисного повідомлення, в якому вказав 120 000 WETH, які потрібно викарбувати. Зловмисник викликав функцію "complete_wrapped", яка зчитує дані з облікового запису шкідливого повідомлення і карбує 120 000 wETH.

Першопричиною цього експлойту є те, що в процесі перевірки ("verify_signatures") програма використовувала застарілу функцію "load_current_index". Ця функція не перевіряє, чи введений "обліковий запис sysvar" насправді є "системним sysvar", що дозволяє зловмиснику фальсифікувати цей критичний обліковий запис.

Експлойт Wormhole Bridge є суворим нагадуванням про потенційні вразливості, притаманні DeFi-протоколам, і про витонченість атак в DeFi-просторі. Першопричиною експлоіту стала невдала валідація облікових записів "опікунів", що дозволило зловмиснику викарбувати 120 000 WETH без жодного підкріплення ETH.

Цей інцидент підкреслює важливість ретельного аудиту смарт-контрактів і потенційні пастки певних шаблонів дизайну при розробці смарт-контрактів. Він також підкреслює витонченість атак в просторі DeFi, де зловмисники використовують складну взаємодію між декількома протоколами.

Оскільки використання цих технологій продовжує зростати, вкрай важливо врахувати ці уроки, щоб запобігти подібним інцидентам у майбутньому. Цей інцидент слугує нагадуванням про потенційні вразливості протоколів DeFi і необхідність вжиття надійних заходів безпеки, включаючи використання контрактів з декількома підписами, регулярний аудит і розробку надійних протоколів відновлення. Він також підкреслює необхідність постійних досліджень і розробок для виявлення і усунення потенційних вразливостей в технології блокчейн і протоколах DeFi.

Висновки за розділом 2

У даному розділі були проаналізовані найпоширеніші атаки на розумні контракти в блокчейні Ethereum і історичні інциденти в яких вони були реалізовані. Ось кілька найпоширеніших типів атак на смарт-контракти:

Атаки повторного входу: Ця атака відбувається, коли зовнішній контракт перехоплює потік управління і повторно входить в викликаючий контракт в тій же точці, що призводить до потенційного багаторазового виведення коштів. Найвідомішим прикладом є атака DAO, під час якої було викрадено Ефіру на суму близько 50 мільйонів доларів.

Цілочисельні переповнення і недоповнення: Це прості арифметичні помилки, коли число перевищує максимальну ємність змінної (переповнення) або опускається нижче її мінімальної ємності (недоповнення), що призводить до неочікуваних результатів.

Атаки на короткі адреси: Якщо адресні дані коротші, ніж повинні бути, віртуальна машина Ethereum (EVM) не повідомляє про це як про помилку, а замість цього замінює короткі дані нулями. Щоб уникнути цих проблем, дуже важливо дотримуватися найкращих практик розробки смарт-контрактів, ретельно перевіряти код і, можливо, навіть страхувати контракти від потенційних втрат.

РОЗДІЛ 3
РОЗРОБКА РЕКОМЕНДАЙЦІЙ ПО ПІДВИЩЕННЮ РІВНЯ ЗАХИЩЕНОСТІ БЛОКЧЕЙН ДОДАТКІВ

3.1 Рекомендації при розробці блокчейн додатків

При розробці блокчейн-додатків важливо визначити мету проекту, вивчити технологію блокчейн, забезпечити безпеку даних та ефективність системи, розробити зручний інтерфейс і інтегрувати з криптовалютами. Також слід проводити тестування та валідацію, розглядати механізми голосування та управління, а також залучати досвідчених фахівців у розробці. Смарт-контракти, програмовані транзакції, що лежать в основі технології блокчейн, мають значні вразливості. Ці вразливості підкреслюють важливість дотримання найкращих практик під час розробки смарт-контрактів. У даній роботі викладено найважливіші рекомендації щодо розробки смарт-контрактів, підкреслюючи, чому ці найкращі практики мають вирішальне значення для захисту цифрових активів і забезпечення цілісності блокчейн-платформ. Смарт-контракти - самодостатні контракти, умови яких безпосередньо записані в коді, - зробили революцію в транзакціях на блокчейні. Однак автономна і незмінна природа смарт-контрактів також створює значні ризики. Оскільки ці контракти часто оперують величезними сумами та конфіденційною інформацією, вразливості в системі безпеки можуть призвести до значних фінансових та репутаційних збитків. Таким чином, впровадження найкращих практик у розробці смарт-контрактів має важливе значення для зменшення потенційних ризиків.

Проведений аналіз атак на блокчейн додатки та інцидентів пов`язаних з ними дозволили розробити наступні рекомендації:

Розробники повинні застосовувати підхід, заснований на тестуванні, пишучи тести для всіх можливих умов перед розгортанням смарт-контракту. Такий підхід не тільки допомагає виявити і виправити вразливості на ранніх стадіях, але й гарантує,

що контракт поводитиметься так, як очікується, в різних сценаріях.Перед розгортанням смарт-контракти повинні пройти ретельний аудит та експертну оцінку. Кілька пар очей можуть допомогти виявити потенційні вразливості, які могли бути пропущені первинними розробниками.Розробники повинні використовувати встановлені бібліотеки контрактної розробки і дотримуватися прийнятих стандартів, таких як ті, що надаються спільнотою Ethereum. Ці бібліотеки і стандарти пройшли ретельне тестування і в цілому вважаються безпечними.Формальна верифікація - це математичний підхід до перевірки правильності програми, і це потужний інструмент для забезпечення того, щоб смарт-контракт поводився так, як передбачалося. Це може бути ефективним способом виявлення потенційних дірок в безпеці або логічних помилок, які можуть бути використані.Впровадження цих кращих практик може зіграти вирішальну роль в підвищенні безпеки смарт-контрактів. Дотримання цих рекомендацій гарантує стійкість контрактів до відомих вразливостей, підвищує цілісність блокчейн-платформ і вселяє довіру серед користувачів. Крім того, використання найкращих практик може призвести до економії коштів за рахунок мінімізації ймовірності фінансових втрат через вразливості контрактів.

Безпечна розробка смарт-контрактів має вирішальне значення в цифровому ландшафті, що розвивається. Дотримання найкращих практик може суттєво зменшити ризик атак, тим самим

3.1.1 Регулярна перевірка та аудит коду.

Перевірка та аудит коду є важливими практиками для забезпечення безпеки та цілісності смарт-контрактів. Вони передбачають ретельне вивчення коду контракту досвідченими розробниками або аудиторами безпеки для виявлення вразливостей, потенційних зловживань і помилок кодування. Ось кілька прикладів поширених проблем, виявлених під час перевірки та аудиту коду, а також можливі шляхи їх вирішення:

Атаки на повернення: Атака повторного входу відбувається, коли контракт викликає зовнішній контракт до завершення власних змін стану, що дозволяє зовнішньому контракту знову увійти в початковий контракт і потенційно маніпулювати його станом. Щоб запобігти атакам зворотного входу, дотримуйтесь шаблону "перевірки-ефекти-взаємодія". Переконайтеся, що виклики зовнішніх функцій здійснюються після завершення всіх змін стану та переказу коштів.

Цілочисельне переповнення/недоповнення: Цілочисельна арифметика може призвести до неочікуваної поведінки, коли значення перевищують максимальні або мінімальні межі типу даних. Ці вразливості можуть бути використані для маніпулювання станом контракту або крадіжки коштів. Щоб запобігти цьому, використовуйте безпечні математичні бібліотеки, такі як OpenZeppelin[22] SafeMath або Solidity SafeMath, які надають функції для безпечних арифметичних операцій.

Атаки на відмову в обслуговуванні (DoS): Смарт-контракти можуть бути вразливими до DoS-атак, якщо вони не призначені для обробки великих або дорогих обчислень. Наприклад, зловмисник може створити транзакцію, яка споживає надмірну кількість газу, перешкоджаючи обробці інших транзакцій. Щоб запобігти DoS-атакам, обмежте цикли та обчислення до розумного і відомого максимуму, використовуйте ліміти газу та встановіть автоматичні вимикачі або обмеження на основі часу для дорогих операцій.

Вразливості контролю доступу: Неправильно налаштовані або відсутні механізми контролю доступу можуть призвести до несанкціонованого доступу та маніпуляцій з функціями та даними контракту. Щоб вирішити цю проблему, впровадьте систему контролю доступу на основі ролей (RBAC) або систему управління дозволами, щоб обмежити доступ до критично важливих функцій і даних. Використовуйте принцип найменших привілеїв і ретельно переглядайте та тестуйте логіку контролю доступу.

Атаки на випередження: Атаки на випередження відбуваються, коли зловмисник використовує часову затримку між подачею транзакції та її підтвердженням у блокчейні. Він може маніпулювати порядком транзакцій на свою користь. Щоб запобігти випереджувальним атакам, використовуйте такі методи, як

схеми фіксації та розкриття транзакцій, шифрування або впровадження механізмів, подібних до стандарту EIP-3074, який спрямований на запобігання випереджувальним атакам в Ефіріумі.

Неперевірені зовнішні виклики: Зовнішні виклики ненадійних контрактів без належної перевірки можуть створювати ризики для безпеки. Шкідливі або погано реалізовані контракти можуть використовувати вразливості і поставити під загрозу безпеку вашого смарт-контракту. Завжди перевіряйте вихідний код і репутацію зовнішніх контрактів, перш ніж звертатися до них. Розгляньте можливість використання білих або чорних списків контрактів для посилення заходів безпеки.

Міркування щодо газових лімітів: Смарт-контракти повинні працювати в межах ліміту газу в блоці. Обтяжливі в обчислювальному плані контракти або контракти з неефективним кодом можуть перевищити ліміт газу, що призведе до збоїв у транзакціях або потенційних зловживань. Оптимізуйте код вашого контракту, зменшіть кількість газоємних операцій і проведіть ретельне тестування використання газу, щоб забезпечити сумісність з газовими обмеженнями мережі.

Для усунення цих та інших потенційних вразливостей рекомендується залучати зовнішніх аудиторів безпеки або фірми, які спеціалізуються на аудиті смарт-контрактів. Такі аудитори можуть провести глибокий аналіз і тестування вашого коду, виявити вразливості і надати рекомендації щодо підвищення безпеки.

Важливо відзначити, що перегляд коду і аудит повинні бути постійними процесами протягом усього життєвого циклу розробки, а не просто одноразовими заходами. Регулярно оновлюйте контракти, переглядайте зміни та проводьте аудит, щоб реагувати на нові ризики безпеки та впроваджувати найкращі практики.

3.1.2 Використання стандартних захищених бібліотек

Стандартні бібліотеки безпеки надають попередньо перевірені і протестовані в боях реалізації різних функцій контрактів, знижуючи ризик впровадження вразливостей і підвищуючи загальну безпеку смарт-контрактів. Ось кілька

прикладів широко використовуваних стандартних бібліотек безпеки і рішень на їх основі:

OpenZeppelin: OpenZeppelin - це популярна бібліотека з відкритим вихідним кодом, яка пропонує повний набір безпечних компонентів для смарт-контрактів, які можна використовувати повторно. Вона забезпечує реалізацію контролю доступу, стандартів токенів (наприклад, ERC20, ERC721), безпечних математичних операцій, операцій, заснованих на часі, і багато іншого. Використання OpenZeppelin допомагає гарантувати, що критичні функціональні можливості реалізовані безпечно і відповідають найкращим практикам.

Бібліотека контрактів ConsenSys: ConsenSys розробила бібліотеку контрактів, яка забезпечує безпечну та перевірену реалізацію різних стандартів та функцій контрактів. Вона включає в себе модулі для контролю доступу, стандарти токенів, можливість оновлення та багато іншого. Бібліотека проходить регулярний аудит безпеки, щоб забезпечити високу якість і надійність коду.

DappHub: DappHub пропонує колекцію бібліотек смарт-контрактів з відкритим вихідним кодом, які зосереджені на безпеці, модульності та багаторазовому використанні. Їхні бібліотеки охоплюють такі сфери, як контроль доступу, математичні операції, функціональність, що базується на часі, та інтеграція з оракулами. Ці бібліотеки добре задокументовані та пройшли перевірку безпеки.

Бібліотеки SafeMath: Бібліотеки SafeMath забезпечують безпечну реалізацію арифметичних операцій для запобігання цілочисельним вразливостям переповнення та неповного заповнення. Прикладами є бібліотека SafeMath, що входить до складу OpenZeppelin, та вбудована бібліотека SafeMath у Solidity. Використовуючи ці бібліотеки, ви можете безпечно виконувати математичні операції та уникати неочікуваної поведінки через арифметичні помилки.

Бібліотеки ERC20 та ERC721: Ці бібліотеки надають стандартизовані реалізації для популярних стандартів токенів ERC20 (взаємозамінні токени) і ERC721 (не взаємозамінні токени). Використання цих бібліотек забезпечує сумісність, безпеку та дотримання стандартів токенів, зменшуючи ризик вразливостей, характерних для функціональності токенів.

Бібліотеки аутентифікації: Бібліотеки аутентифікації, такі як Keyvaulth, uPort або OAuth2, забезпечують безпечні та стандартизовані методи аутентифікації та авторизації користувачів в рамках смарт-контрактів. Ці бібліотеки обробляють складні криптографічні операції, безпечно керують приватними ключами та забезпечують інтеграцію з постачальниками ідентифікаційних даних або протоколами автентифікації.

Використовуючи стандартні бібліотеки безпеки, розробники можуть скористатися знаннями та досвідом широкої спільноти і отримати вигоду від добре перевірених, надійних і безпечних реалізацій коду. Ці бібліотеки проходять регулярні перевірки безпеки та оновлення для усунення нових загроз і вразливостей. Однак дуже важливо вивчити і зрозуміти код і документацію цих бібліотек, щоб переконатися, що вони відповідають конкретним вимогам вашого смарт-контракту і мережі.

3.1.3 Тестування блокчейн додатків

Тестування є критично важливим аспектом розробки смарт-контрактів для забезпечення коректності, функціональності та безпеки коду. Ось кілька прикладів тестів, які зазвичай виконуються при розробці смарт-контрактів, а також потенційні рішення:

Модульні тести: Модульні тести перевіряють окремі функції або компоненти смарт-контракту ізольовано. Вони допомагають виявити помилки, граничні ситуації і гарантують, що кожна функція поводиться так, як очікується. Наприклад, ви можете написати модульні тести для перевірки правильності математичних операцій, логіки контролю доступу або передачі токенів. Такі інструменти, як Truffle, Hardhat або Solidity, надають засоби для написання та виконання модульних тестів.

Інтеграційні тести: Інтеграційні тести перевіряють взаємодію та сумісність між різними компонентами або контрактами. Вони гарантують, що контракти працюють разом за призначенням і що система функціонує правильно в цілому.

Наприклад, ви можете протестувати взаємодію між контрактом на токени і контрактом на краудсейл, щоб перевірити процес покупки і розподілу токенів.

Тести безпеки: Тести безпеки спрямовані на виявлення вразливостей і потенційних уразливостей в смарт-контрактах. Вони включають в себе виконання різних симуляцій атак і оцінку стійкості контракту до поширених загроз безпеки. Приклади включають тестування на атаки типу "повторний вхід", цілочисельне переповнення/недоповнення, фронт-раннінг і атаки типу "відмова в обслуговуванні" (DoS). Такі інструменти, як MythX, Securify або Manticore, можуть допомогти в проведенні аналізу безпеки смарт-контрактів.

Тести використання газу: Тести на використання газу гарантують, що смарт-контракти працюють в межах визначених лімітів газу на платформі блокчейну. Вони допомагають виявити газоємні операції або неефективний код, який може призвести до збою транзакцій через помилки, пов'язані з нестачею газу. Проведення тестів на використання газу має вирішальне значення для оптимізації роботи контрактів і мінімізації витрат на газ. Такі інструменти, як Truffle, Hardhat або Remix IDE можуть надати можливості профілювання та оцінки використання газу.

Тести граничних сценаріїв: Граничні тести оцінюють поведінку контракту в екстремальних або неочікуваних сценаріях. Вони включають тестування граничних умов, лімітів і виняткових ситуацій, які можуть виникнути під час виконання контракту. Приклади включають тестування на максимальну кількість токенів, обробку дробових сум токенів або тестування поведінки контракту при збої зовнішніх залежностей.

Тести, що залежать від часу: Тести, що залежать від часу, перевіряють поведінку контракту в часі, особливо коли контракти передбачають операції, що залежать від часу, такі як графіки наділення правами або контроль доступу в часі. Ці тести гарантують, що контракт функціонує правильно і виконує очікувані дії в різні моменти часу. Такі інструменти, як Ganache або Hardhat, надають функціональні можливості для імітації та маніпулювання часом під час тестування.

Тести на зовнішню залежність: Якщо ваш контракт залежить від зовнішніх сервісів, таких як оракули або зовнішні API, важливо включити тести, які

перевіряють інтеграцію та поведінку контракту при взаємодії з цими залежностями. Ці тести допомагають переконатися, що контракт може безпечно і коректно обробляти зовнішні дані або події.

Щоб підвищити ефективність тестування, розгляньте можливість використання фреймворків автоматизованого тестування, конвеєрів безперервної інтеграції (CI) та інструментів аналізу тестового покриття. Ці інструменти допомагають автоматизувати процес тестування, регулярно запускати тести та вимірювати покриття коду, щоб забезпечити комплексний набір тестів.

Важливо відзначити, що тестування має бути безперервним процесом протягом усього життєвого циклу розробки. Регулярно оновлюйте і розширюйте свій набір тестів у міру розвитку контракту, а також повторно запускайте тести після внесення змін, щоб переконатися, що нові доповнення або модифікації коду не призведуть до регресу або вразливостей.

А також чітке розуміння повного спектру загроз які можуть вплинути на роботу додатку і підвищення обізнаності користувачів.

3.2 Модель загроз для блокчейн додатку

В ході дослідження була сформована модель загроз в якій розглянуто п`ять категорій загроз:

- випадкові загрози
- загрози конфіденційності
- загрози фінансового шахрайства
- соціальні загрози
- технічні загрози.

Кожна категорія загроз далі розбита на конкретні загрози з описом агентів загроз, можливих наслідків і контрзаходів для кожної загрози. Модель надає систематизований огляд загроз, з якими можуть зіткнутися користувачі криптовалют, і може бути використана для розуміння ситуацій, в яких користувачі

можуть опинитися під тиском атаки. В кінцевому підсумку це може допомогти в розробці більш безпечних систем.Модель загроз відображена в таблиці 3.1.

Таблиця 3.1

Модель загроз блокчейн додатку

ВИД	ЗАГРОЗА	АГЕНТ	НАСЛІДКИ	КОНТРЗАХОДИ
Випадкові загрози	Неправильний запис облікових даних доступу	Людина (ненавмисно)	Повна втрата криптовалюти	Негайна перевірка облікових даних доступу, таких як мнемоніка, після їх запису. Дизайн додатків може підтримувати цей процес, вимагаючи такої перевірки.
	Поломка обладнання	Людські (ненавмисні), природні	Повна втрата криптовалюти	Користувачі-початківці можуть використовувати надійні кастодіальні платформи, які передбачають механізми відновлення облікових записів. Користувачам, які знають, як керувати ключами, слід створювати резервні копії своїх ключів.
	Помилкова транзакція	Людина (ненавмисно)	Часткова втрата криптовалюти	Контрзаходи: Користувачі повинні ретельно перевіряти кожну транзакцію перед її здійсненням.
Загрози конфіденційності	Деанонімізація	Організована злочинність та злочинці	Викриття персональних даних	Користувачі можуть зменшити ризик деанонімізації, якщо не будуть публічно ділитися криптовалютними адресами.
	Атака "розпилення"	Організована злочинність та злочинці, корпорації	Викриття персональних даних	Жертви можуть або заморозити UTXO, отримані в результаті атаки, або перевести всі UTXO, які не були викрадені, на новий гаманець.
	Атака зіпсованих монет	Організована злочинність та злочинці	Пошкодження репутації, часткові збитки	Зберігання інформації про користувача в таємниці є критично важливим для запобігання цій атаці.
	Крадіжка особистих даних	Організована злочинність та злочинці, люди (навмисні)	Розкриття персональних даних	Криптовалюту можна купити через P2P-біржі, які не вимагають процедури KYC.
Загрози фінансового шахрайства	Pump & Dump	Організована злочинність та злочинці	Девальвація.	Всебічне розуміння ризиків маніпулювання ринком на нерегульованих ринках є критично важливим, особливо для осіб, зацікавлених в інвестуванні в криптовалюти. В ідеалі ці інвестиції повинні бути довгостроковими і

				технологічно орієнтованими.
	Short & Distort	Організована злочинність та злочинці	Девальвація.	Див. контрзаходи щодо викачування та скидання.
	Коротке/довге полювання	Організована злочинність і злочинці, корпорації.	Девальвація.	Уникати централізованих бірж і спекулятивної торгівлі.
	Змити і повторити	Фізичні особи (навмисні), корпорації, організована злочинність і злочинці.	Девальвація.	Утриматися від спекулятивної торгівлі (як зазначено вище)
	Фальшиві стіни	Фізичні особи (навмисні), корпорації, організована злочинність і злочинці.	Девальвація.	Утриматися від спекулятивної торгівлі (як зазначено вище).
Соціальні загрози:	Шахрайство:	Організована злочинність та злочинці.	Повна втрата криптовалюти, репутаційні втрати.	Навчання користувачів щодо оцінки легітимності претензій та розуміння поширених типів загроз соціальної інженерії.
	Фішингові атаки	Організована злочинність і злочинці, нецільові.	Повна втрата криптовалюти, розголошення персональних даних.	Скептично ставитися до небажаної комунікації з боку платформ та інвестувати в освіту користувачів. Використовуйте надійні розширення для браузерів для додаткового захисту.
	Ризик платформи:	Корпорації, працівники.	Повна втрата криптовалюти, тимчасова втрата криптовалюти, розкриття персональних даних.	Користувачі не повинні покладатися на одну платформу, завжди створювати резервні копії своїх криптовалютних ключів і мати право власності на них.
Технічні загрози:	Програми-вимагачі	Організована злочинність і злочинці, не пов'язані з конкретною ціллю	Повна втрата криптовалюти, розкриття персональних даних	Для гаманців для зберігання двофакторна автентифікація забезпечує додаткову безпеку, якщо пристрій скомпрометовано.
	Шахрайські клієнтські	Організована злочинність і	Повна втрата криптовалюти,	Користувачі повинні перевіряти надійність програмного

	програми	злочинці, не пов'язані з конкретною ціллю	розкриття персональних даних	забезпечення гаманців перед використанням.
	Атаки на сторонні сервіси:	Організована злочинність і злочинці, не пов'язані з конкретною ціллю	Повна втрата криптовалюти, розкриття персональних даних	Користувачі повинні вивчити і зрозуміти заходи безпеки, які застосовуються біржами, перш ніж користуватися ними. Великі біржі можуть мати страхові поліси, що покривають втрату коштів клієнтів.
	Загрози смарт-контрактів	Організована злочинність і злочинці, людина (навмисна), людина (ненавмисна)	Часткова втрата криптовалюти	Користувачі повинні переконатися, що смарт-контракти пройшли ретельний аудит безпеки, проведений авторитетними фірмами.
	Атаки на транзакції:	Організована злочинність і злочинці, люди (навмисні)	Часткова втрата криптовалюти, тимчасова втрата криптовалюти	Уникайте інвестування в невідомі криптовалюти.

Модель загроз поділяється на п`ять категорій:

Випадкові загрози: Ці загрози виникають через людські помилки, несправність обладнання або стихійні лиха. Вони не є навмисними, але можуть призвести до значних втрат.

Випадкові загрози - це ризики, які виникають через ненавмисні людські помилки або недогляди, несправності обладнання або стихійні лиха. Ці загрози не пов'язані з навмисними діями зловмисника. До цієї категорії можна віднести наступні загрози:

1. Неправильний запис облікових даних доступу: Це відбувається, коли облікові дані доступу, такі як паролі, мнемоніки або приватні ключі, записані неправильно, що робить гаманець і пов'язані з ним криптовалюти недоступними в майбутньому.

Агенти загроз: Людина (ненавмисно)

Наслідки: Повна втрата криптовалюти

Контрзаходи: Негайна перевірка облікових даних доступу, таких як мнемоніка, після їх запису. Дизайн додатків може підтримувати цей процес, вимагаючи такої перевірки.

2. Втрата облікових даних: Ця загроза виникає, коли облікові дані доступу, такі як паролі, приватні ключі, мнемоніки та інші форми резервних копій, правильно записані, але неналежним чином зберігаються, що призводить до їх можливої втрати. Неналежне зберігання може включати в себе нездатність зберігати облікові дані доступу або не враховувати збої в роботі обладнання чи катастрофи. Можна виділити наступні підкатегорії:

Забуття облікових даних доступу: Сюди входить забування паролів до гаманця, брелоків до гаманця або місцезнаходження пристрою зберігання, якщо він зберігається в "секретному" місці.

Випадкове знищення: Це трапляється, коли користувачі випадково знищують облікові дані, наприклад, перезаписують файл wallet.dat, форматують жорсткий диск або викидають носій інформації.

Поломка обладнання: Це трапляється, коли обладнання, на якому зберігаються облікові дані, виходить з ладу через технічну несправність, і немає доступних вторинних резервних копій.

Руйнівні катастрофи: Втрата облікових даних внаслідок стихійних лих або "непереборної сили", таких як пожежа, повінь або падіння метеорита.

Всі ці підкатегорії мають наступні характеристики:

Агенти загроз: Людські (ненавмисні), природні

Наслідки: Повна втрата криптовалюти

Контрзаходи: Користувачі-початківці можуть використовувати надійні кастодіальні платформи, які передбачають механізми відновлення облікових записів. Користувачам, які знають, як керувати ключами, слід створювати резервні копії своїх ключів. Професійні користувачі, які оперують великими сумами, можуть розглянути можливість використання розширеної інфраструктури та сторонніх провайдерів для додаткової безпеки.

3. Помилкова транзакція: Також відомі в розмовній мові як транзакції "товстого пальця" або "золотого пальця" - це помилки, допущені при виконанні транзакції. Можна виділити наступні підкатегорії:

Неправильно вказана адреса: Вводиться неправильна, але дійсна адреса одержувача, в результаті чого транзакція надсилається на недійсну або іноземну адресу без можливості скасувати її.

Неправильно вказана сума: Це трапляється, коли вводиться неправильна сума, в результаті чого на адресу призначення надсилається сума, більша за заплановану.

Неправильно вказана комісія: Це трапляється, коли вводиться неправильна комісія за транзакцію. Комісія стягується з майнера без можливості її відшкодування.

Всі ці підкатегорії мають наступні характеристики:

Агенти загроз: Людина (ненавмисно)

Наслідки: Часткова втрата криптовалюти

Контрзаходи: Користувачі повинні ретельно перевіряти кожну транзакцію перед її здійсненням. Розробники повинні створювати користувацькі інтерфейси таким чином, щоб легко відстежувати транзакції з "товстими пальцями".

Загрози конфіденційності: Ці загрози передбачають співставлення даних про публічні транзакції з інформацією з додаткових джерел, таких як соціальні мережі або витоки даних, з метою отримання персональних даних про жертву.

Загрози конфіденційності стосуються ризиків, пов'язаних із співставленням даних про публічні транзакції та інформації з додаткових джерел, таких як соціальні мережі або витоки даних, з метою отримання персональних даних про користувача криптовалюти. Хоча ці загрози не можуть безпосередньо призвести до втрати криптовалюти, вони можуть полегшити подальші атаки. До цієї категорії можна віднести наступні загрози:

1. Деанонімізація: Це передбачає аналіз існуючих цифрових артефактів, таких як транзакції або соціальні мережі, з метою ідентифікації віртуальної або реальної особи або компанії, що володіє криптовалютою. Ця інформація може бути використана для подальших атак.

Агенти загроз: Організована злочинність та злочинці

Наслідки: Викриття персональних даних

Контрзаходи: Користувачі можуть зменшити ризик деанонімізації, якщо не будуть публічно ділитися криптовалютними адресами, використовуватимуть криптовалюти, орієнтовані на конфіденційність, або користуватимуться сервісами змішування криптовалют. Однак повне уникнення деанонімізації вимагає глибокого розуміння властивостей конфіденційності різних криптовалют.

2. Атака "розпилення" (Dusting Attack): Ця атака полягає у надсиланні невеликих сум криптовалюти на велику кількість криптовалютних адрес. Спостерігаючи за подальшими транзакціями, зловмисник може співвіднести різні адреси гаманців, контрольовані одним користувачем, з метою пов'язати "запилені" адреси з особою власника.

Агенти загроз: Організована злочинність та злочинці, корпорації

Наслідки: Викриття персональних даних

Контрзаходи: Жертви можуть або заморозити UTXO, отримані в результаті атаки, або перевести всі UTXO, які не були викрадені, на новий гаманець. Захист від атак вимагає значної обізнаності про залишки на своїх рахунках.

3. Атака зіпсованих монет: У цій атаці зловмисник свідомо переказує криптовалюту, отриману злочинним шляхом, жертві, прагнучи пов'язати жертву та адресу її гаманця зі злочином. Це може призвести до того, що наявні у жертви монети стануть менш взаємозамінними, а сама жертва стане об'єктом кримінального розслідування.

Агенти загрози: Організована злочинність та злочинці

Наслідки: Пошкодження репутації, часткові збитки

Контрзаходи: Зберігання інформації про користувача в таємниці є критично важливим для запобігання цій атаці. Після ураження забруднені монети можуть бути повернуті відправнику або використані сервіси змішування для очищення забруднених монет.

4. Крадіжка особистих даних: Ця загроза виникає через політику "Знай свого клієнта" (Know-Your-Customer, KYC), яка вимагає від кастодіальних бірж збирати

реальну ідентифікаційну інформацію від клієнтів. Інформація, розкрита біржі або сторонньому провайдеру KYC, є цінною мішенню для зловмисників.

Агенти загроз: Організована злочинність та злочинці, люди (навмисні)

Наслідки: Розкриття персональних даних

Контрзаходи: Криптовалюту можна купити через P2P-біржі, які не вимагають процедури KYC. Для централізованих бірж зменшення кількості інформації, що передається, може знизити ризик..

1. Загрози фінансового шахрайства: Ці загрози пов'язані з систематичним маніпулюванням криптовалютними ринками, яке може відбуватися через їхню нерегульовану природу.

Ризики фінансового шахрайства на ринках криптовалют виникають через відсутність регуляторного нагляду. Такі шахрайські дії можуть безпосередньо не призвести до втрати криптовалюти, натомість вони знецінюють активи жертви. Ця вразливість поширюється на всі нерегульовані ринки, і навіть регульовані ринки, такі як фондовий ринок, можуть бути вразливими, хоча нормативно-правові акти часто забороняють такі дії. Нижче розглядаються наступні загрози цієї категорії:

Pump & Dump: Ця схема завищує ціну криптовалюти, створюючи ажіотаж у соціальних мережах. Після того, як достатня кількість жертв купує завищену криптовалюту, зловмисники продають свої частки, що призводить до падіння ціни.

Агенти загроз: Організована злочинність та злочинці.

Наслідки: Девальвація.

Контрзаходи:

Всебічне розуміння ризиків маніпулювання ринком на нерегульованих ринках є критично важливим, особливо для осіб, зацікавлених в інвестуванні в криптовалюти. В ідеалі ці інвестиції повинні бути довгостроковими і технологічно орієнтованими. Адекватна обізнаність про потенційні ризики є найкращим превентивним заходом.

Щоб обійти схеми "pump & dump", дуже важливо бути обізнаним і утримуватися від імпульсивних дій з купівлі або продажу.

Short & Distort: Ця схема передбачає маніпулювання ринком, щоб викликати падіння цін шляхом поширення негативних чуток у соціальних мережах. Шахраї отримують прибуток за рахунок шортування криптовалюти перед атакою.

Агенти загроз: Організована злочинність та злочинці.

Наслідки: Девальвація.

Контрзаходи: Див. контрзаходи щодо викачування та скидання.

Коротке/довге полювання: Біржі зі значними активами можуть впливати на коливання цін, купуючи/продаючи самі себе. Ця маніпуляція може спричинити ліквідацію коротких/довгих позицій, що є фінансовим стимулом для бірж, оскільки вони отримують прибуток від торгових зборів.

Агенти загрози: Організована злочинність і злочинці, корпорації.

Наслідки: Девальвація.

Контрзаходи: Уникати централізованих бірж і спекулятивної торгівлі.

Змити і повторити: криптовалютні "кити" або суб'єкти, що контролюють значні суми криптовалюти, можуть спровокувати раптові стрибки цін. Вони можуть створювати заявки на продаж нижче ринкової ціни, викликаючи падіння цін і панічні продажі. Потім вони викуповують криптовалюту за нижчою ціною, отримуючи прибуток.

Агенти загрози: Фізичні особи (навмисні), корпорації, організована злочинність і злочинці.

Наслідки: Девальвація.

Контрзаходи: Утриматися від спекулятивної торгівлі (як зазначено вище).

Фальшиві стіни: Кити можуть сфабрикувати великі замовлення на купівлю або продаж, створюючи "стіну", яка впливає на цінову траєкторію. Наступні користувачі слідують за трендом, виставляючи вищі/нижчі ордери на купівлю/продаж. Однак "кити" можуть скасувати свої замовлення і виконати вищі/нижчі замовлення, розміщені жертвами.

Агенти загрози: Фізичні особи (навмисні), корпорації, організована злочинність і злочинці.

Наслідки: Девальвація.

Контрзаходи: Утриматися від спекулятивної торгівлі (як зазначено вище).

Інсайдерська торгівля: За відсутності регуляторних гарантій, інсайдери можуть використовувати привілейовану непублічну інформацію. Наприклад, працівники великих бірж або творці токенів можуть використовувати інформацію про майбутні лістинги на популярній біржі, щоб отримати вигоду від підвищення цін після публічних оголошень.

Агенти загроз: Фізичні особи (навмисні), корпорації.

Наслідки: Девальвація.

1. Соціальні загрози: Ці загрози експлуатують соціальну природу людини, наприклад, її довіру до інших людей та організацій.

Соціальні ризики експлуатують довіру жертв. Їх можна розділити на дві категорії: Соціальна інженерія, яка передбачає психологічну маніпуляцію, щоб переконати людей виконати певні дії або розкрити конфіденційну інформацію, та Платформні ризики, коли зловживають довірою до сторонньої платформи. У межах цієї категорії існують наступні загрози:

Шахрайство: Шахрайство охоплює всі загрози, які обманом змушують користувача розподіляти ресурси, такі як фіатні гроші або криптовалюта, з шахрайською метою. Ця загроза також поділяється на:

Шахрайський обмін (Exit Scam): Це стосується бірж або гаманців, створених з наміром викрасти криптовалюту користувача на більш пізньому етапі.

Шахрайська криптовалютна афера: Ці шахрайства заманюють велику кількість жертв інвестувати в нібито криптовалюту на основі шахрайських обіцянок. Прикладами є фінансові піраміди, фальшиві ICO і фальшиві криптовалюти, названі на честь існуючих компаній або проектів.

Шахрайство з транзакціями: Ці шахраї обманом змушують жертву надсилати криптовалюту, не надаючи натомість обіцяних послуг. Приклади включають продаж фальшивих токенів, локальні продажі біткоїнів і шахрайські продавці, які не можуть доставити обіцяні товари.

Шахрайство з видачею себе за іншу особу: Ці шахрайські схеми змушують жертву думати, що відома або заможна особа роздає криптовалюту безкоштовно.

Жертву обманом змушують відправити криптовалюту на адресу зловмисника, вважаючи, що вона отримає назад більше, ніж відправила.

Шахрайство з шантажем: Цей тип шахрайства змушує користувача думати, що зловмисник володіє конфіденційною інформацією про жертву, яку він погрожує оприлюднити, якщо не заплатити викуп.

Всі ці види шахрайства мають наступні характеристики загрози:

Агенти загрози: Організована злочинність та злочинці.

Наслідки: Повна втрата криптовалюти, репутаційні втрати.

Контрзаходи: Навчання користувачів щодо оцінки легітимності претензій та розуміння поширених типів загроз соціальної інженерії. Уникати пропозицій, які здаються занадто хорошими, щоб бути правдою, або створюють відчуття терміновості. Використовуйте розширення для браузерів, такі як EtherAddressLookup, для додаткового захисту.

Фішингові атаки: Фішингові атаки - це атаки, які обманом змушують користувача розголошувати конфіденційну інформацію, наприклад, паролі або приватні ключі, зловмиснику. Зловмисники використовують копії легітимних на вигляд веб-сайтів, наприклад, бірж, щоб обманом змусити користувачів надати свої облікові дані. До цієї категорії загроз належать

Фішинг електронної пошти: зловмисники надсилають електронні листи, видаючи себе за надійне джерело, з метою викрадення особистої інформації.

Рекламний фішинг: зловмисники використовують рекламу в пошукових системах та/або соціальних мережах, щоб перенаправити жертву на фішинговий сайт.

Фішинг у соціальних мережах: зловмисники використовують прямі повідомлення в соціальних мережах або на приватних форумах, щоб перенаправити жертву на фішинговий сайт.

Голосовий фішинг: включає в себе атаки соціальної інженерії за допомогою телефонних дзвінків, де зловмисники видають себе за світові бренди та довірені агенції.

SMS-фішинг (SMiShing): Зловмисники використовують текстові повідомлення на мобільні телефони, щоб спонукати жертву до негайних дій.

Spear-фішинг: передбачає цілеспрямовану фішингову атаку на окремих власників криптовалют з метою отримання контролю над їхніми криптовалютами.

Всі ці атаки мають наступні характеристики загрози:

Агенти загрози: Організована злочинність і злочинці, нецільові.

Наслідки: Повна втрата криптовалюти, розголошення персональних даних.

Контрзаходи: Скептично ставитися до небажаної комунікації з боку платформ та інвестувати в освіту користувачів. Використовуйте надійні розширення для браузерів для додаткового захисту. Для кастодіальних бірж користувачі повинні отримувати доступ до платформи безпосередньо через свою URL-адресу та увімкнути двофакторну автентифікацію за допомогою безпечної парольної фрази. Для більш просунутих користувачів додаткову безпеку можуть забезпечити рішення для холодного зберігання даних.

Ризик платформи: це ризик, пов'язаний з централізованими платформами, такими як біржі або гаманці для зберігання, які можуть не відповідати місцевим законам і правилам, тим самим обмежуючи людей в доступі, відправленні або отриманні криптовалют. Такі платформи можуть закрити або заблокувати обліковий запис, обмежити можливості здійснення транзакцій, заборонити іншим користувачам платформи надсилати транзакції на певну адресу або відмовити в доступі до ключів певного облікового запису.

Агенти загроз: Корпорації, працівники.

Наслідки: Повна втрата криптовалюти, тимчасова втрата криптовалюти, розкриття персональних даних.

Контрзаходи: Користувачі не повинні покладатися на одну платформу, завжди створювати резервні копії своїх криптовалютних ключів і мати право власності на них.

1. Технічні загрози: Ці загрози виникають через технології, що використовуються для взаємодії з криптовалютними системами.

Наступні загрози пов'язані з технологіями, що використовуються в криптовалютних системах. Ми зосередимося на загрозах на рівні додатків, які безпосередньо впливають на взаємодію користувача з системою. Ми не будемо розглядати загрози, пов'язані з базовим рівнем інфраструктури, рівнем консенсусу або конкретними реалізаціями криптовалют. В рамках цієї категорії можна виділити наступні загрози:

Шкідливе програмне забезпечення: Шкідливе програмне забезпечення - це шкідливе програмне забезпечення, яке працює в системі жертви без її відома і має на меті отримати доступ до її активів або криптовалют. Цю загрозу можна розділити на підтипи:

- Шкідливе програмне забезпечення для викрадення гаманців/ключів: Це шкідливе програмне забезпечення викрадає приватні ключі або сховище гаманця (наприклад, файл "wallet.dat") з системи жертви для подальшого шифрування.

- Шкідливе програмне забезпечення для маніпулювання транзакціями: Це шкідливе програмне забезпечення маніпулює окремими транзакціями, перенаправляючи їх на адреси, контрольовані зловмисником. Наприклад, "Перехоплювач буфера обміну" прослуховує скопійовані адреси криптовалют і замінює їх на адресу зловмисника.

- Шкідливе програмне забезпечення для викрадення облікових даних: Це шкідливе програмне забезпечення викрадає облікові дані користувача, наприклад, кейлоггери, що перехоплюють паролі, введені на платформах обміну криптовалют, таких як Coinbase.

- Програми-вимагачі: Вимагачі шифрують дані жертви, включаючи її гаманець, і вимагають викуп за розшифровку.

Ці підвиди мають наступні характеристики загрози:

- Агенти загрози: Організована злочинність і злочинці, не пов'язані з конкретною ціллю

- Наслідки: Повна втрата криптовалюти, розкриття персональних даних

- Контрзаходи:

- Для гаманців для зберігання двофакторна автентифікація забезпечує додаткову безпеку, якщо пристрій скомпрометовано.
- Користувачі програмних гаманців, підключених до Інтернету (гарячі гаманці), повинні використовувати безпечні паролі.
- Великі кошти, особливо ті, що зберігаються протягом тривалого періоду, слід переміщувати на холодні гаманці.
- Гаманці повинні мати надійні резервні копії, бажано не на одному пристрої.

Транзакції слід ретельно перевіряти на точність перед відправленням. Розробники гаманців повинні полегшити ці перевірки для користувачів (наприклад, перевірка адреси та суми).

Шахрайські клієнтські програми: Шахрайські клієнтські програми роблять вигляд, що пропонують послуги користувачам, але при цьому таємно маніпулюють вихідними даними на користь зловмисника. Ця загроза включає наступні підвиди:

- Шахрайський генератор ключів/гаманців: Це можуть бути апаратні або програмні засоби, які генерують гаманець для користувача, але також надають зловмиснику доступ до приватних ключів, часто шляхом їх попереднього обчислення. Жертва не знає, що зловмисник може отримати доступ до криптовалют, які зберігаються в гаманці.

- Шахрайський гаманець: Шахрайські гаманці маскуються під безпечний клієнтський додаток для управління криптовалютами. Воно або надсилає приватні ключі користувача зловмиснику при імпорті існуючого гаманця, або маніпулює транзакціями за лаштунками.

- Шахрайський генератор/сканер QR-кодів: Ця загроза пов'язана з генераторами або сканерами QR-кодів, які маніпулюють закодованою адресою одержувача, замінюючи її на адресу зловмисника.

Ці підвиди мають наступні характеристики загрози:

- Агенти загроз: Організована злочинність і злочинці, не пов'язані з конкретною ціллю

- Наслідки: Повна втрата криптовалюти, розкриття персональних даних

- Контрзаходи:

- Користувачі повинні перевіряти надійність програмного забезпечення гаманців перед використанням.
- Програмне забезпечення гаманців слід завантажувати лише з перевірених джерел та перевіряти на цілісність.
- QR-коди слід сканувати або генерувати безпосередньо за допомогою надійних гаманців, уникаючи сторонніх додатків.

Атаки на сторонні сервіси: Ці атаки спрямовані на сервіси, на які покладаються користувачі криптовалют, а не на їхні персональні пристрої. Ця загроза включає в себе наступні підвиди:

- Злом онлайн-бірж: Зловмисники компрометують криптовалютну біржу або гаманець, що призводить до тимчасової недоступності, часткової втрати або повної втрати криптовалют, якими керують користувачі. Атака на біржу часто призводить до того, що біржа оголошує про банкрутство, що ускладнює відновлення коштів для користувачів.

- Маніпуляції з блокчейн-платформами: Маніпулювання платформами для дослідження блоків, які надають інтерфейси для перевірки стану блокчейну (наприклад, Etherscan). Жертви можуть бути обмануті, вважаючи, що транзакція відбулася, хоча насправді її не було, що слугує сходинкою в скоординованих атаках.

- Атаки з підміною SIM-карт: Зловмисники маніпулюють телеком-провайдерами, щоб перенести номер телефону жертви на власну SIM-карту. Ця техніка часто використовується при захопленні облікових записів для обходу двофакторної автентифікації.

Ці підвиди мають наступні характеристики загроз:

- Агенти загроз: Організована злочинність і злочинці, не пов'язані з конкретною ціллю

- Наслідки: Повна втрата криптовалюти, розкриття персональних даних

- Контрзаходи:

- Користувачі повинні вивчити і зрозуміти заходи безпеки, які застосовуються біржами, перш ніж користуватися ними. Великі біржі можуть мати страхові поліси, що покривають втрату коштів клієнтів.

- Доступ до веб-дослідників блоків повинен здійснюватися через безпечні TLS-з'єднання, а користувачі повинні перевіряти дійсні сертифікати. Перехресна перевірка транзакцій за допомогою різних дослідників блоків може допомогти виявити маніпуляції.
- Користувачі можуть захиститися від атак на підміну SIM-карт, захистивши свої телекомунікаційні акаунти надійними паролями. Для двофакторної автентифікації можна використовувати додатки-автентифікатори, а не покладатися на SMS-повідомлення.

Загрози смарт-контрактів: Ці загрози виникають при взаємодії зі смарт-контрактами. Користувачі можуть не знати, що мають справу зі смарт-контрактом, особливо при роботі з токенами ERC20, реалізованими на блокчейні Ethereum. Існують наступні підформи:

- Бекдор для адміністратора: Смарт-контракти навмисно містять бекдори, які дозволяють привілейованим користувачам виводити кошти. Ці бекдори часто хитро приховані за допомогою побічних ефектів програмування, які не відразу помітні при перевірці коду.

- Honeypot-контракти: Контракти Honeypot обманом змушують користувачів надсилати кошти, обіцяючи витік довільному користувачеві (жертві). Однак кошти, надані користувачем, стають пасткою, доступною лише зловмиснику.

- Ненавмисні вразливості смарт-контрактів: Смарт-контракти можуть містити технічні вразливості, які дозволяють зловмисникам отримати доступ до коштів або спричиняють неочікувану поведінку, що призводить до втрат. Класифікація поширених вразливостей є активною сферою досліджень.

Ці підвиди мають наступні характеристики загроз:

- Агенти загроз: Організована злочинність і злочинці, людина (навмисна), людина (ненавмисна)

- Наслідки: Часткова втрата криптовалюти

- Контрзаходи:

- Користувачі повинні переконатися, що смарт-контракти пройшли ретельний аудит безпеки, проведений авторитетними фірмами.

- Перевірений вихідний код контракту повинен бути доступний на таких платформах, як Etherscan для смарт-контрактів Ethereum, і користувачі повинні перепровіряти код.

Атаки на транзакції: Атаки на транзакції передбачають маніпуляції з транзакціями в блокчейні. Наступні підтипи представляють найпоширеніші загрози:

- Атака на більшість (51% атак): Зловмисники отримують контроль над більшістю ресурсів, задіяних в механізмі консенсусу, що дозволяє їм маніпулювати минулими транзакціями. Ці атаки більш реалістичні для менш популярних криптовалют.

- Подвійні витрати: Зловмисники транслюють транзакцію в блокчейн, а потім проводять іншу транзакцію з вищою комісією, яка переводить ті ж самі кошти на іншу адресу, що перебуває під їхнім контролем. Друга транзакція замінює початкову, що призводить до її збою.

- Флуд-атака: Зловмисники генерують велику кількість транзакцій, переповнюючи портфель очікуваних транзакцій (mempool) і викликаючи затримки в підтвердженні інших транзакцій. Це призводить до неочікуваного часу очікування для кінцевих користувачів.

- Інші атаки базового рівня: Окремі криптовалюти можуть бути вразливими до додаткових загроз, спрямованих на рівень консенсусу, рівень інфраструктури (наприклад, DDoS-атаки, NTP-атаки) або мережевий рівень (наприклад, атаки на маршрутизацію і розбиття мережі). Ці загрози вимагають незалежного дослідження, що виходить за рамки цього проекту.

Ці підтипи мають наступні характеристики загроз:

- Агенти загроз: Організована злочинність і злочинці, люди (навмисні)
- Наслідки: Часткова втрата криптовалюти, тимчасова втрата криптовалюти
- Контрзаходи:
    - Уникайте інвестування в невідомі криптовалюти.
    - Дочекайтеся рекомендованої кількості підтверджень після включення транзакції в блокчейн, перш ніж вважати її успішною.

Висновки за розділом 3

Розробка та підтримка блокчейн-додатків, особливо тих, що базуються на Ethereum, вимагає комплексного та ретельного підходу. Використання найкращих практик, таких як аудит і рецензування коду, ретельне тестування, обмеження прав доступу, безпечне кодування і використання перевірених бібліотек, є невід'ємною частиною забезпечення надійності і безпеки цих додатків.

Аудит коду та рецензування забезпечують необхідну лінію захисту, виявляючи потенційні вразливості, які могли бути пропущені під час розробки. Регулярне тестування, як автоматизоване, так і ручне, дозволяє розробникам виявляти і виправляти проблеми до розгортання, знижуючи ризик атак. Обмеження прав доступу, життєво важливий принцип безпечного кодування, гарантує, що тільки авторизовані суб'єкти мають доступ до критично важливих функцій, що ще більше зміцнює структуру безпеки програми.

Використання довірених бібліотек - ще одна рекомендація, якою не можна нехтувати. Ці бібліотеки, які, як правило, вважаються безпечними завдяки їх широкому тестуванню, забезпечують міцну основу для створення смарт-контрактів. Крім того, безпечні методи кодування підвищують стабільність додатків і знижують ймовірність вразливостей, які можуть бути використані.

Поряд з цими кращими практиками, розробка комплексної моделі загроз для блокчейн-додатків має вирішальне значення. Добре побудована модель загроз дозволяє розробникам проактивно виявляти, аналізувати та усувати потенційні загрози. Це дозволяє їм випереджати нові виклики безпеці та гарантувати, що їхні додатки оснащені засобами захисту від атак.

Оскільки технологія блокчейн продовжує розвиватися, а її впровадження зростає, ці практики стануть все більш важливими у формуванні безпечного та надійного цифрового майбутнього. У цьому розділі наведено рекомендації для розробки та підтримки блокчейн-додатків на базі Ethereum, а саме: аудит та рецензування коду, тестування, та обмеження дозволів, безпечне кодування та

використання перевірених бібліотек. Також була розроблена модель загроз для блокчейн-додатку.

ВИСНОВКИ

У даній кваліфікаційній роботі було проведено дослідження технології блокчейн,проведено аналіз атак та інцидентів на розумні контракти засновані на блокчейні Ethereum, розроблено модель загроз для блокчейн додатків та розроблено рекомендації з підвищення рівня захисту розумних контрактів .

У першій частині було детально розглянуто архітектуру блокчейн систем, їх внутрішню структуру та найбільш розповсюджені типи контрактів. Досліджено способи використання,функції розумних контрактів та проаналізовані їх ключові відмінності.

У другому розділі було досліджено атаки та інциденти пов`язані з розумними контрактами на платформі Ethereum. Було розглянуто такі атаки, як атака повторного входу,атака відтворення, атака на цілочисельне переповнення та інші.За результатами аналізу інцидентів було виділено 2 основні загрози які були використані на практиці для проведення атак на блокчейн дотатки-це атака повторного входу і фішинг.

У третьому розділі була створена комплексна модель загроз які виникають при використанні додатків заснованих на технології блокчейн в мережі Ethereum. На основі проаналізованих інцидентів та грунтуючись на моделі загроз були сформовані рекомендації з підвищення рівня захищеності додатків заснованих на розумних контрактах. Ці рекомендації дозволять виявляти потенційні проблеми з безпекою в додатках та аналізувати їх рівень захищеності та запобігати потенційним загрозам.

Виходячи із поставленої мети дипломної роботи були виконані наступні завдання:

- Проведено аналіз інцидентів повязаних з компроментацією блокчейн технологій;

- Досліджено типові вразливості віртуальної машини Ethereum та оцінено ризики її експлуатації;
- Розроблено модель загроз для сучасних блокчейн додатків;
- Розроблено рекомендації щодо підвищення рівня захищеності блокчейн додатків

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. What is Ethereum? [Електронний ресурс]. – Режим доступу: https://ethereum.org/en/what-is-ethereum/

2. Документація Ethereum [Електронний ресурс]. – Режим доступу: https://github.com/ethereum/solidity#documentation

3. What is a smart contract, and how does it work? [Електронний ресурс]. – Режим доступу: https://cointelegraph.com/learn/what-are-smart-contracts-a-beginners-guide-to-automated-agreements

4. Solidity Patterns [Електронний ресурс]. – Режим доступу: https://fravoll.github.io/solidity-patterns/

5. Smart Contracts in Finance: The Arrow is Up [Електронний ресурс]. – Режим доступу: https://hedera.com/learning/smart-contracts/smart-contracts-finance

6. Multiple-Layer Security Threats on the Ethereum Blockchain and Their Countermeasures [Електронний ресурс]. – Режим доступу: https://www.hindawi.com/journals/scn/2022/5307697/

7. Brown, J. (2018). Ethereum smart contract security. [Електронний ресурс]. – Режим доступу: https://medium.com/@jeremybrown/ethereum-smart-contract-security-e8c8e0747a7d

8. Preventing Smart Contract Attacks on Ethereum — Reentrancy attack [Електронний ресурс]. – Режим доступу: https://betterprogramming.pub/preventing-smart-contract-attacks-on-ethereum-a-code-analysis-bf95519b403a

9. 10 Key Smart Contract Vulnerabilities: That can Lock your Crypto Assets [Електронний ресурс]. – Режим доступу: https://www.immunebytes.com/blog/smart-contract-vulnerabilities/

10. Learn security risks with a new honeypot scam [Електронний ресурс]. – Режим доступу: https://goplussecurity.medium.com/learn-security-risks-with-a-new-honeypot-scam-36bdbf772aa3

11. What Was The DAO? [Електронний ресурс]. – Режим доступу: https://www.gemini.com/cryptopedia/the-dao-hack-makerdao

12. Bancor smart contracts vulnerability: It’s not over [Електронний ресурс]. – Режим доступу: https://zengo.com/bancor-smart-contracts-vulnerability-its-not-over/

13. Detailed disclosure and repair plan for the “False top-up” vulnerability in the Ethereum token [Електронний ресурс]. – Режим доступу: https://slowmist.medium.com/detailed-disclosure-and-repair-plan-for-the-false-top-up-loopholes-in-the-ethereum-token-952f4aa748a2

14. DEPOSafe: Demystifying the Fake Deposit Vulnerability in Ethereum Smart Contracts [Електронний ресурс]. – Режим доступу: https://www.researchgate.net/publication/342120914_DEPOSafe_Demystifying_the_Fake_Deposit_Vulnerability_in_Ethereum_Smart_Contracts

15. Known Attacks [Електронний ресурс]. – Режим доступу: https://ethereum-contract-security-techniques-and-tips.readthedocs.io/en/latest/known_attacks/

16. Smart contract security [Електронний ресурс]. – Режим доступу: https://ethereum.org/az/developers/docs/smart-contracts/security/

17. Oracle Manipulation [Електронний ресурс]. – Режим доступу: https://consensys.github.io/smart-contract-best-practices/attacks/oracle-manipulation/

18. A survey of attacks on Ethereum smart contracts Nicola Atzei, Massimo Bartoletti, and Tiziana Cimoli Universit`a degli Studi di Cagliari, Cagliari, Italy

19. Most Common Smart Contract Vulnerabilities and How to Prevent Them [Електронний ресурс]. – Режим доступу: https://pixelplex.io/blog/smart-contract-vulnerabilities/

20. Threat Modeling for the Blockchain [Електронний ресурс]. – Режим доступу: https://www.howardposton.com/blog/threat-modeling-for-the-blockchain

21. Serhan W. Bahar Advanced Security Threat Modelling for

Blockchain-Based FinTech Applications [Електронний ресурс]. – Режим доступу: https://arxiv.org/ftp/arxiv/papers/2304/2304.06725.pdf

22. OpenZeppelin. (2020). OpenZeppelin: The secure smart contract development framework. [Електронний ресурс]. – Режим доступу:https://openzeppelin.com/

23. Explained: The bZx Hack [Електронний ресурс]. – Режим доступу: https://www.halborn.com/blog/post/explained-the-bzx-hack-november-2021

24. Wormhole Bridge Exploit Incident Analysis [Електронний ресурс]. – Режим доступу: https://www.certik.com/resources/blog/1kDYgyBcisoD2EqiBpHE5l-wormhole-bridge-exploit-incident-analysis

25. What is a Blockchain Protocol Audit? [Електронний ресурс]. – Режим доступу: https://academy.paidnetwork.com/lesson/what-is-a-blockchain-protocol-audit

26. Threat Modeling for the Blockchain [Електронний ресурс]. – Режим доступу: https://www.howardposton.com/blog/threat-modeling-for-the-blockchain

27. The Ultimate Guide to Blockchain Oracles [Електронний ресурс]. – Режим доступу: https://worldcoin.org/articles/what-is-an-oracle-in-blockchain

28. Defining Smart Contract Defects on Ethereum [Електронний ресурс]. – Режим доступу: Jiachi Chen; Xin Xia; David Lo; John Grundy; Xiapu Luo; Ting Chen https://ieeexplore.ieee.org/abstract/document/9072659

29. Smart Contracts Categorization With Topic Modeling Techniques Giacomo Ibba/ Marco Ortu, Roberto - Tonelli University of Cagliari, Department of Mathematics and Computer Science, Cagliari, Italy

30. Ways to Ensure Smart Contract Security [Електронний ресурс]. – Режим доступу: https://www.leewayhertz.com/ways-to-ensure-smart-contract-security/